Facultad de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Herramienta para evaluar el daño producido en el ADN espermático de centollo a través del análisis de células cometa

Estudiante: Diego Suárez García
Dirección: Mª Noelia Barreira Rodríguez
Andrés Martínez Lage

A Coruña, febreiro de 2020.

Dedicado a mi Yo del pasado, del presente y del futuro.

Agradecimientos

Este proyecto no hubiera sido posible sin la ayuda y profesionalidad de Mª Noelia Barreira Rodríguez. Desde el primero hasta el último día se ha ofrecido a ayudarme con sus conocimientos y sobre todo interés. Es por ello que le doy las gracias. Agradecer también a mi otro director Andrés Martínez Lage por estar disponible cuando lo necesitaba y haberme brindado la oportunidad de hacer un proyecto de mi interés. Gracias a mi familia y a todas las personas que compartieron momentos felices conmigo en estos años. Finalmente, gracias al usuario de la comunidad *Stack Overflow*, *theGtknerd*, por haberme dedicado su tiempo ayudándome con el desarrollo gráfico.

Resumen

El ácido desoxirribonucleico o ADN es el contenedor de las instrucciones con las que nuestros organismos se guían durante nuestra etapa evolutiva como seres vivos. Desde finales de siglo XX, los expertos utilizan el *ensayo del cometa* para estudiar el daño producido en el ADN. Su estudio es importante porque proporciona información de qué agentes físicos y químicos externos pueden deteriorar su estructura. El ensayo del cometa se hizo rápidamente popular porque es sencillo, económico y accesible. Un equipo de biólogos de la Universidade da Coruña está realizando un estudio sobre el daño presente en el ADN de células espermáticas de centollo empleando el ensayo del cometa. Del ensayo obtienen imágenes microscópicas que necesitan analizar para obtener resultados cuantitativos. Para ello emplean herramientas automatizadas que segmentan el ADN en las imágenes y computan las métricas necesarias. No obstante, las herramientas gratuitas para el análisis de imágenes del ensayo no procesan adecuadamente sus imágenes mientras que las herramientas comerciales son costosas. Este proyecto consiste en el desarrollo de una herramienta que cubra sus necesidades. La solución propuesta se basa en la detección y segmentación del ADN en las imágenes mediante el uso de algoritmos automáticos de procesado de imagen por medio del estudio del histograma de las imágenes y la integración de dicha metodología en una herramienta que permite además la segmentación manual de las imágenes de forma simple y eficaz.

Abstract

Since the end of the 20th century, experts worldwide have been widely using the comet assay to study the damage caused to the DNA. Its analysis is relevant because it can reveal what physical and chemical agents have induced deterioration in its structure. The comet assay became popular due to its simplicity, affordability and accessibility. A team of biologists from the University of A Coruña is studying the damage on spider crab spermatic cells DNA using the comet assay. From the assay they obtain microscopic images that need further analysis to retrieve quantitative results. However, available free tools have not been able to process successfully their images and the commercial tools are too expensive. This project involves the development of a tool to fulfill their needs. The solution consists of image processing algorithms to detect and segment the DNA structures through the study of the image histogram which are integrated in a tool that allows the manual segmentation of the images in a simply and effective way.

Palabras clave:

- células
- ADN
- ensayo cometa
- detección de objetos
- segmentación en imágenes
- metodología ágil
- Scrum
- Python
- GTK
- Cairo

Keywords:

- cells
- DNA
- comet assay
- object detection
- image segmentation
- agile methodology
- Scrum
- Python
- GTK
- Cairo

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Las centollas hembras (*Maja brachydactyla*), al igual que otros crustáceos, tienen unas estructuras denominadas *espermatecas* donde acumulan los espermatozoides después de su apareamiento con los machos. La cópula se produce normalmente en aguas profundas del litoral a las que migran los individuos durante el otoño. Una vez que han copulado vuelven a aguas someras en las que realizan las puestas a lo largo del año. En Galicia suelen realizar hasta tres puestas al año empleando el esperma acumulado, siendo la última hacia los meses de verano. Esto implica que el esperma puede permanecer en las espermatecas durante 8 e incluso 10 meses. En base a las características de las células espermáticas de los centollos se cree que el daño espermático en la segunda y tercera puesta es mayor con respecto a la primera y, por lo tanto, la viabilidad de los embriones o larvas es menor. Por consiguiente, se quiere realizar un estudio para comprobar si hay más daño genético en las últimas puestas que en las primeras. Para ello se emplea el *ensayo del cometa*, una técnica que evalúa el daño presente en el ADN de las células.

1.1 Contextualización

El ensayo de electroforesis alcalina de células individuales, más comúnmente conocido como *ensayo del cometa*, es utilizado para evaluar el daño producido en el ADN de las células por diferentes agentes físicos y químicos externos [1]. El ensayo del cometa se lleva empleando de forma extensa desde su primera introducción en 1984 [2]. Cuatro años más tarde se publicó la versión estándar que se conoce actualmente [3].

El proceso consiste en sumergir los núcleos de las células en gel de agarosa, romper su membrana (lisis) y someterlos a un campo eléctrico (electroforesis alcalina). Los núcleos sanos mantienen su estructura a pesar de los efectos del campo eléctrico. Los núcleos dañados, sin embargo, sufren la migración de parte de su material genético en dirección al ánodo del campo eléctrico. Esto ocurre de forma directamente proporcional a lo dañado que se encuen-

tre su ADN. Es aquí cuando los núcleos suficientemente dañados dejan impresa esa silueta que asemeja a la de un cometa del sistema solar, donde el núcleo sería la cabeza del cometa, y los fragmentos del material genético que han migrado constituirían la cola. Es por ello que el ensayo recibe este característico nombre y los núcleos el de *cometas*. Para hacerlos visibles, los cometas se tiñen con un colorante específico para ADN al final del proceso. En la Figura 1.1 se puede ver de forma más representativa el procedimiento del ensayo del cometa. Obsérvense los dos cometas de ejemplo resultantes. El cometa de la derecha es un núcleo dañado que ha sufrido migración de su material genético. El cometa de la izquierda ha mantenido su estructura porque no está dañado.

El éxito del ensayo del cometa radica en su asequibilidad económica, su simplicidad, sensibilidad, versatilidad y rapidez. El número de células necesarias para llevar a cabo este ensayo es relativamente muy bajo en comparación a la disponibilidad celular individual [4].

El ensayo del cometa se emplea en múltiples campos en el ámbito de la medicina como toxicología medioambiental [5], biomonitorización de poblaciones [6], radiación biológica [7], estudios nutricionales y estudios cancerígenos [8][9]. En general, cualquier proceso que involucre potencial daño en el ADN.

1.2 Estado del arte

De los núcleos tratados por el ensayo se obtienen imágenes microscópicas (Fig. 1.7) que son analizadas para medir el daño producido en el ácido desoxirribonucleico. Para estudiar el daño se analiza la relación entre la cabeza y la cola de los cometas. En la Figura 1.2 hay un núcleo sano (izquierda) y otro dañado (derecha). En el núcleo dañado la cabeza está en verde y la cola en rojo. El núcleo sano no tiene forma de cometa y no posee cola. Cuanto mayor es el daño producido, menor cantidad de ADN hay en la cabeza del cometa. Esto se traduce en una cabeza más pequeña con respecto a la cola y en una menor intensidad lumínica.

En un principio, las imágenes se inspeccionaban visualmente; sin embargo, la métrica visual era mayormente cualitativa y estaba sujeta a la subjetividad del experto que la estuviera llevando a cabo [4]. Además, era una tarea tediosa y poco eficiente. Es de esperar que pronto surgieran herramientas con el fin de agilizar el análisis de los resultados de este ensayo y que ofrecieran métricas cuantitativas y universales.

En la actualidad son varias las aplicaciones disponibles para analizar los resultados del ensayo. Una de ellas es *CASPLab* [10]. CASPLab surgió a raíz de una tesis en un máster de física teórica en la universidad de Wroclaw en 2003. Continuó ofreciendo soporte de forma gratuita hasta 2015 y es multiplataforma. CASPLab únicamente soporta imágenes en formato *TIFF*. Estas pueden ser a color o en escala de grises. Los usuarios deben configurar a mano algunos parámetros y seleccionar el área de la imagen que quieren analizar. Cada vez que se

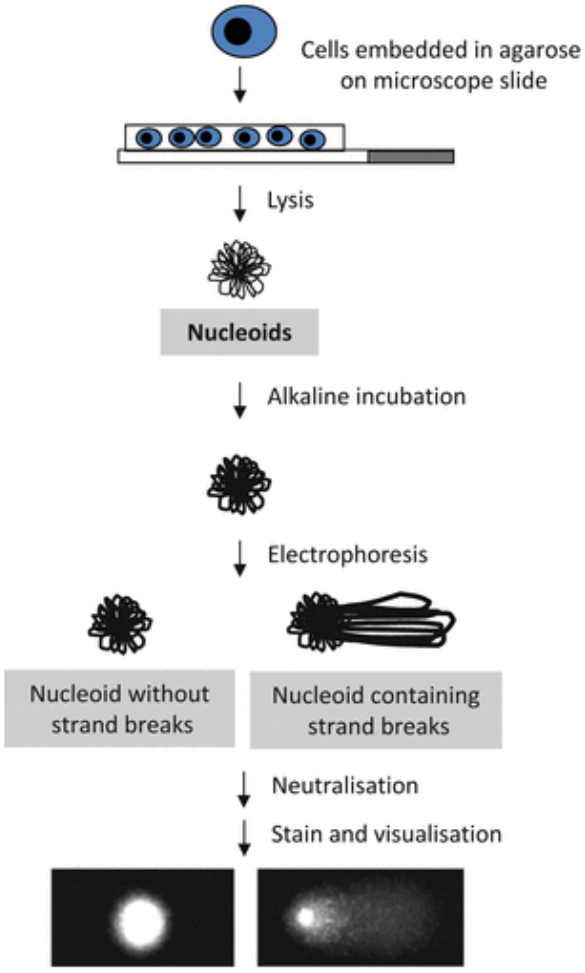


Figura 1.1: Procedimiento estándar del ensayo de electroforesis alcalina. i) Las células son sumergidas en gel de agarosa. ii) Se rompe su membrana (lisis). iii) Se incuban en una disolución alcalina. iv) Se someten a un campo eléctrico (electroforesis alcalina). v) Se neutraliza la disolución y se aplica colorante.

analiza un cometa, el programa almacena los resultados y abre una ventana donde muestra el perfil de intensidad [11] del cometa y otros parámetros. El usuario puede ver en cualquier momento los resultados de los análisis y exportarlos a un archivo de texto. En la Figura 1.3 se muestra una imagen de cómo lucía la interfaz gráfica de CASPLab.

Otro programa bastante popular fue *CometScore* [12]. CometScore fue desarrollado por la compañía especializada en *hardware* TriTek Corp. Originalmente respondía bajo el nombre de *AutoComet* y era un *software* de uso comercial. En la actualidad es gratuito y la última versión estable fue lanzada en noviembre de 2017. Actualmente no dispone de soporte y sólo es compatible con Windows. CometScore sólo soporta imágenes de mapa de bits. Al igual que CASPLab, CometScore requiere que el usuario introduzca algunos parámetros. Por ejemplo, los usuarios deben mover una barra desplazable vertical de la interfaz para que el fondo de la imagen vaya desapareciendo hasta que sólo quedan los cometas visibles en ella. Este fase puede verse en 1.4. En la imagen de la izquierda se ve la imagen que está siendo analizada y a la derecha la imagen después de haber sido eliminado el fondo. Los usuarios disponen de varias formas para analizar los cometas. Una forma es hacer un recuadro alrededor del cometa que se quiere analizar y seleccionar el centro de la cabeza. CometScore se encarga de analizar el cometa con estas entradas. Otra forma es seleccionar múltiples cometas como si de selección de archivos se tratara y dejar que el programa decida cuál es el centro de la cabeza de cada cometa. Un método más automático es dejar que el programa obtenga por él mismo los cometas de la imagen. Finalmente, permite eliminar cometas que no sean de interés para el usuario y exportar los resultados en una hoja de cálculo.

Comet Assay IV [13] se diferencia de los previamente mencionados porque está diseñado para funcionar mientras se obtienen imágenes en tiempo real de los núcleos. Es un programa con soporte actual y de uso comercial. El funcionamiento de este programa es sencillo. El usuario puede desplazarse por las imágenes que está captando la cámara. Cuando hace *click* en un cometa, este es automáticamente analizado. No permite una selección múltiple de cometas. En la Figura 1.5 se puede ver a Comet Assay IV en funcionamiento.

Komet 7 [14] es otro sistema completamente diseñado para trabajar retransmitiendo imágenes en vivo con un equipo integrado. Creado originalmente en 1992 por Kinetic Imaging, sigue actualizándose bajo licencia por las manos de Andor Technology.

OpenComet [15] surgió para paliar los problemas que presentaban los programas gratuitos de su momento. A diferencia de CASPLab y CometScore, OpenComet permite tanto el análisis de imágenes en tiempo real como por separado. Igual que ellos, es completamente gratuito. Se trata de un *plug-in* del programa de procesado de imágenes ImageJ. A diferencia de sus antecesores, soporta los formatos de imágenes más comunes y es multiplataforma. Tanto el análisis de las imágenes de entrada como la segmentación de los cometas y el cómputo de las métricas las realiza de forma automática. En la Figura 1.6 se adjunta una imagen de la inter-

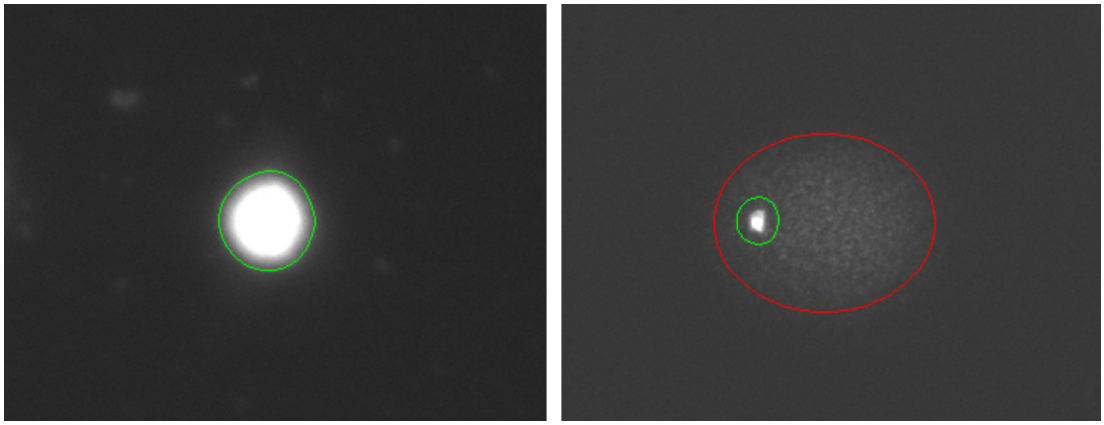


Figura 1.2: Izq.: núcleo sano. Dcha.: núcleo dañado con forma de cometa.

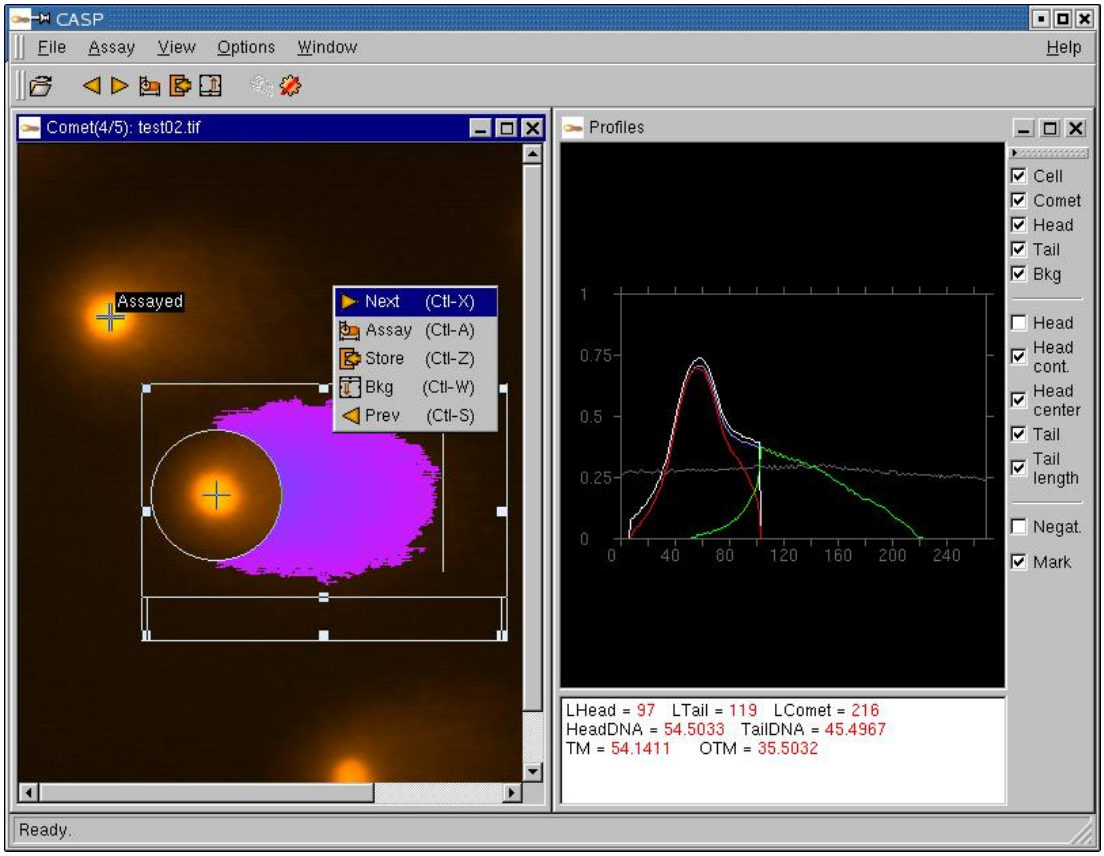


Figura 1.3: Interfaz gráfica de la última versión estable de CASPLab.

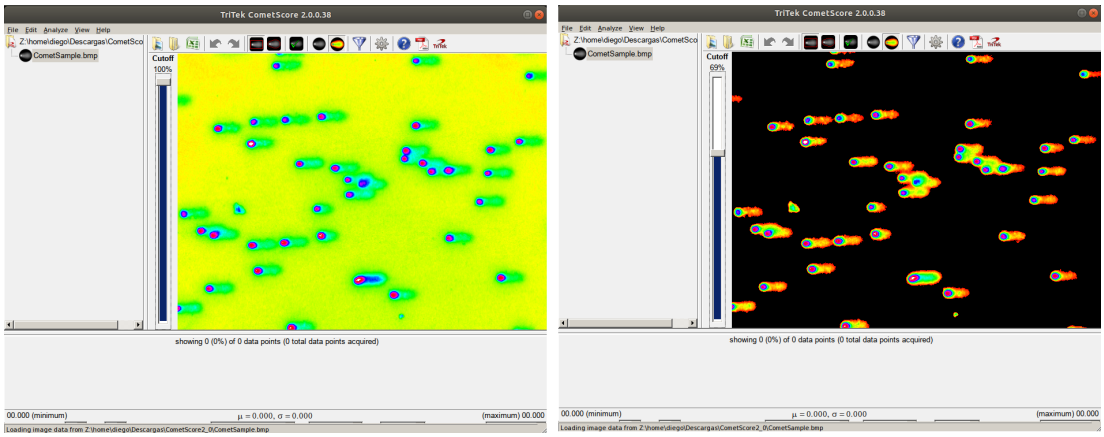


Figura 1.4: Interfaz gráfica de CometScore 2.0. A la derecha se ha manipulado la variable *Cutoff* para quitar el fondo de la imagen.

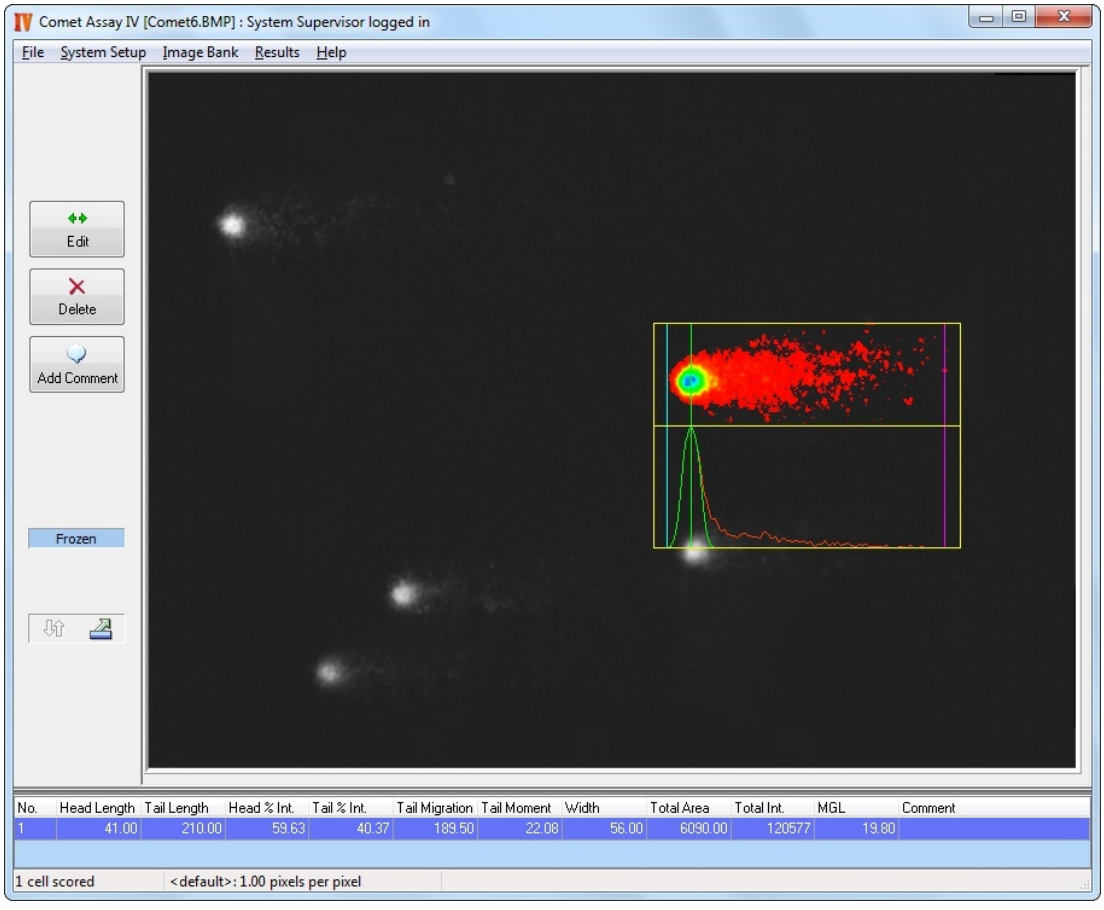


Figura 1.5: Interfaz gráfica de Comet Assay IV con una retransmisión en vivo. Al hacer *click* en un cometa este es analizado automáticamente.

faz de OpenComet. Los usuarios únicamente tienen que seleccionar las imágenes que quieren analizar y escoger el directorio de salida. El ingreso de parámetros por parte del usuario es opcional. Por cada imagen, OpenComet crea otra con la segmentación de los cometas. También genera una hoja de cálculo con las métricas de los cometas de todas las imágenes analizadas.

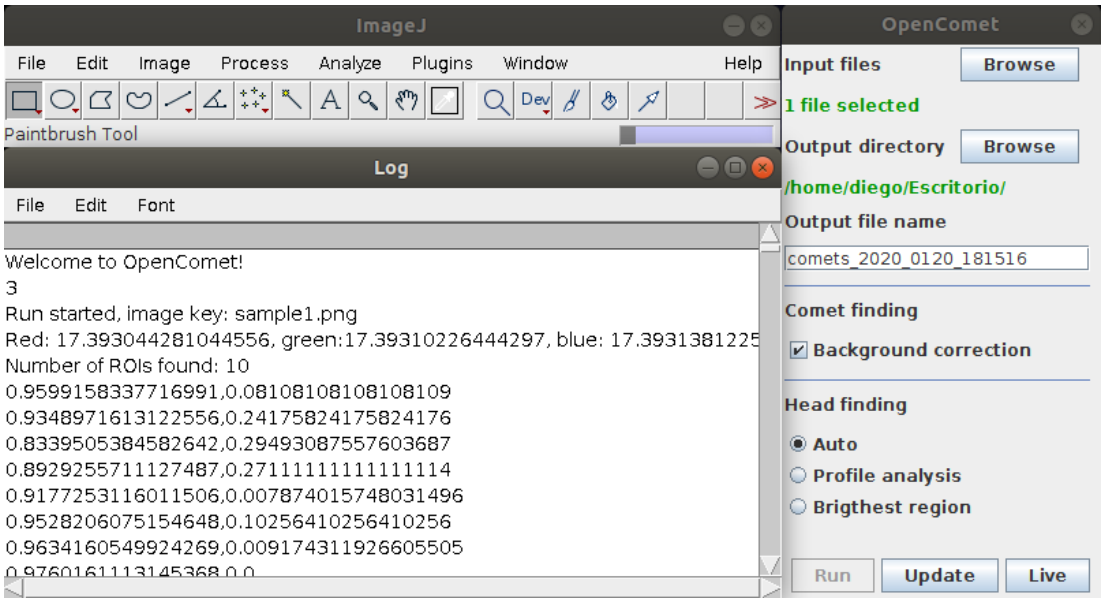


Figura 1.6: Interfaz gráfica de ImageJ con la herramienta de OpenComet en funcionamiento.

Sin embargo, ni OpenComet ni el resto de herramientas son siempre capaces de segmentar los cometas de forma correcta. Esto se debe a que las imágenes que se obtienen del ensayo del cometa pueden llegar a ser significativamente variables (Fig. 1.7). El problema se acentúa a medida que el porcentaje de segmentaciones erróneas se incrementa. Aquí surge la principal problemática del panorama actual. Las opciones que tienen los usuarios de forma gratuita no permiten la creación ni edición de los cometas segmentados. Por otra parte, el resto de programas son de licencia comercial y costosos.

OpenComet se utilizó como herramienta para segmentar las imágenes de las células espermáticas de centollo de este estudio. No obstante, las segmentaciones que se obtuvieron no fueron en la mayoría de los casos correctas. En la Figura 1.8 se muestra una comparativa entre las segmentaciones obtenidas al ejecutar OpenComet (columna izquierda) y las segmentaciones realizadas a mano por un experto (columna derecha). En la fila superior, OpenComet considera que el contorno de la cola de los cometas es prácticamente también el contorno de la cabeza. En la fila inferior acierta con los contornos de las colas pero no con los de las cabezas.

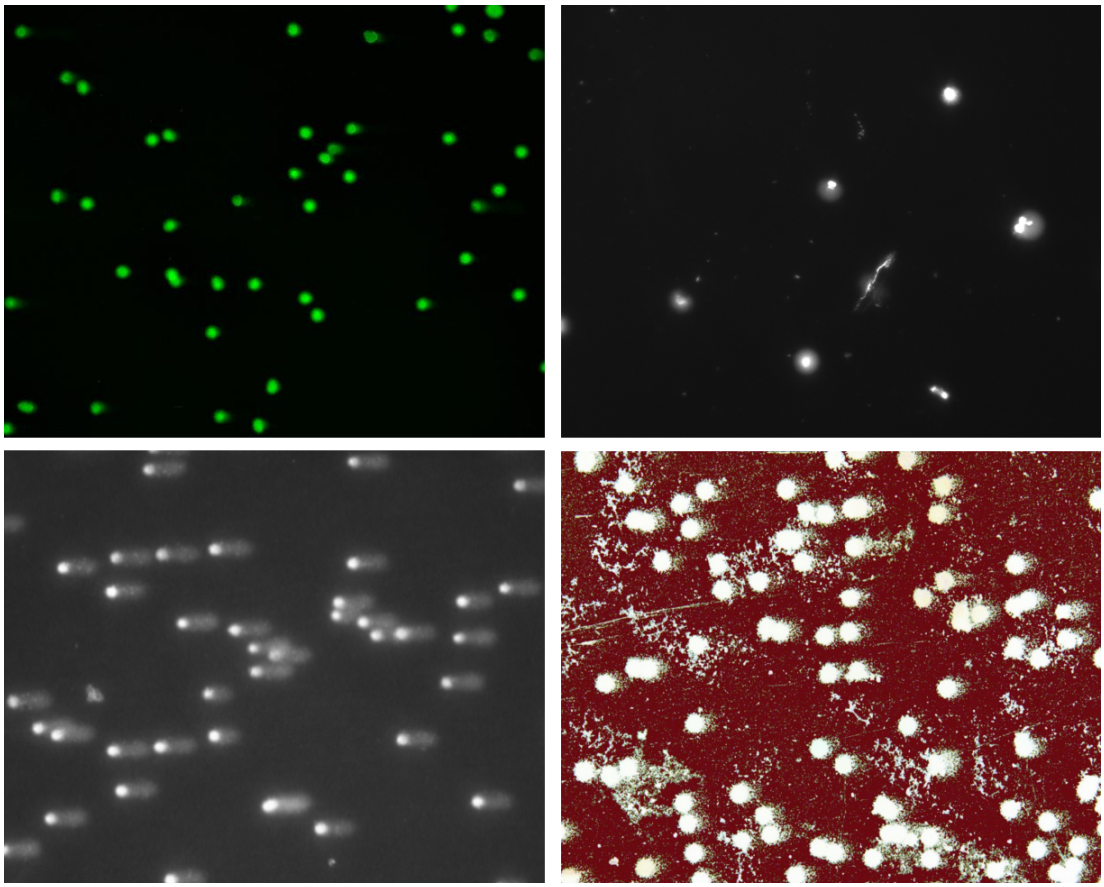


Figura 1.7: Distintas imágenes microscópicas de núcleos de células tratados por el ensayo del cometa.

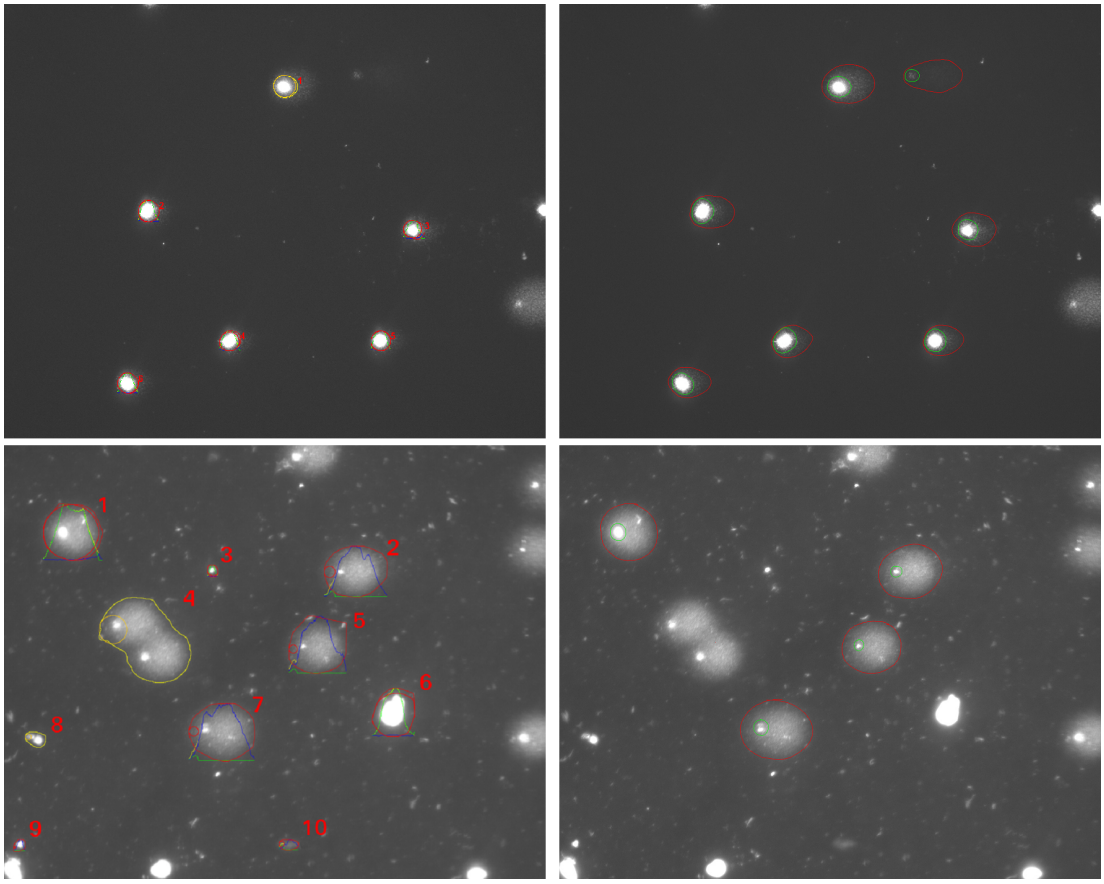


Figura 1.8: Izq.: segmentaciones realizadas automáticamente con OpenComet. Dch.: segmentaciones realizadas manualmente.

1.3 Objetivo

El propósito de este proyecto es proporcionar una solución al estado del arte actual del ensayo del cometa. Para esto, se propone desarrollar una herramienta gratuita y multiplataforma que permita tanto la segmentación automática de imágenes como la segmentación manual. Así, en caso de que las imágenes no sean segmentadas correctamente por la herramienta, los usuarios podrán hacerlo de forma manual.

La herramienta implementará una metodología que segmente imágenes en los formatos más comunes de imagen, tanto a color como a escala de grises. Los usuarios tendrán la posibilidad de modificar de forma sencilla, intuitiva y eficaz los resultados de la segmentación en caso de no estar satisfechos con el criterio de la herramienta. También podrán segmentar ellos mismos sin emplear el analizador automático con las herramientas que ofrecerá la aplicación. Se les permitirá crear y guardar proyectos para guardar su trabajo en memoria y recuperarlo cuando lo consideren conveniente. Los usuarios podrán generar una hoja de cálculo con las métricas estandarizadas de todos los cometas segmentados.

1.4 Estructura de la memoria

A continuación se desglosa en qué capítulos se estructura el resto de la memoria de este trabajo, describiendo brevemente en qué consiste cada uno. Estos capítulos llevarán al lector por el recorrido seguido hasta alcanzar el completo desarrollo del proyecto:

Capítulo 2 - Planificación del proyecto: Se expone cómo se realizó la planificación del proyecto para su desarrollo. Así, se comenta de cómo se ha adaptado a este la metodología de desarrollo empleada, de cómo se ha dividido el proceso de inicio a fin, del seguimiento realizado y de la estimación de costes prevista.

Capítulo 3 - Tecnologías y materiales: Se introducen todas las herramientas que se han empleado a lo largo del proyecto y los materiales con los que se contó para su elaboración. También se adjunta una tabla con las versiones de cada tecnología utilizada.

Capítulo 4 - Metodología de segmentación: Se explica en qué consiste la metodología para la segmentación automática de las imágenes, los procesos por los que se ha transitado hasta su desarrollo y se exponen los resultados obtenidos.

Capítulo 5 - Aplicación: Se describe la construcción de la aplicación desde su análisis hasta el diseño, implementación y pruebas.

Capítulo 6 - Conclusiones y líneas futuras: Se hace un recorrido por los pasos que se han

seguido hasta la elaboración completa del proyecto, se exponen las conclusiones obtenidas tras dicha elaboración y se comentan posibles mejoras.

Capítulo 2

Planificación del proyecto

En este capítulo se expone en qué consiste el marco de trabajo *Scrum* y cómo se ha aplicado en este proyecto. También se explica cómo se ha ejecutado la planificación y seguimiento de este. Finalmente, se detalla la estimación de costes del proyecto.

2.1 Metodología de desarrollo

Para el desarrollo de este proyecto se ha optado por seguir la metodología ágil de desarrollo *Scrum* [16]. En general, como metodología ágil [17], Scrum se ajusta mejor a proyectos pequeños y está pensado para conseguir *software* de calidad en plazos cortos de tiempo con flexibilidad a cambios en los requisitos. Su esencia radica en crear de forma iterativa productos incrementales potencialmente listos para despliegue. Cada uno de estas iteraciones se conoce como *sprint*. La idea principal es poder entregar al cliente el producto resultante de cada sprint para que pueda interactuar, ver el avance del proyecto y dar retroalimentación al equipo. En la Figura 2.1 se resume el proceso global de Scrum.

El equipo de trabajo estuvo compuesto por Mª Noelia Barreira Rodríguez, Andrés Martínez Lage y Diego Suárez García. El rol de *Product Owner* fue ejercido tanto por Andrés como por Noelia, pues ambos proporcionaron los requisitos y necesidades del producto final. El papel de *Scrum Master* lo interpretó Noelia, ejerciendo de moderadora y gestionando los sprints. El *Development Team* estuvo compuesto por Diego, quién creó los productos de cada sprint.

El equipo se reunió en una primera instancia en la que se creó el *Product Backlog* con todos los requisitos que debía cumplir el producto final. El primer sprint se definió en esta misma reunión con las funcionalidades que su finalización debía cubrir.

Se conocía de antemano la imposibilidad del equipo de trabajo a realizar reuniones diarias. No obstante, no fueron esenciales al estar el *Development Team* constituido por una única persona. Por esta razón, estas reuniones fueron sustituidas por correos electrónicos, videollamadas y reuniones en persona esporádicas entre el *Scrum Master* y el *Development Team*.

Al finalizar un sprint el equipo de trabajo se reunía o bien en persona o por videollamada para analizar los resultados y definir el siguiente sprint.

2.2 Planificación

En la primera reunión se determinaron los principales sprints en los que se dividiría el desarrollo del proyecto. En la Figura 2.2 se adjunta un cronograma en el que se muestran los sprints en función del tiempo. Estos se listan a continuación con una breve descripión.

Sprint 1: Análisis del estado del arte

Se pretende profundizar en el estado del arte del ensayo del cometa. Así, este análisis incluye la contextualización del ensayo del cometa, la recopilación y estudio de las herramientas disponibles actuales para analizar los resultados del ensayo y la investigación de las metodologías empleadas en la segmentación de los cometas.

Sprint 2: Implementación de OpenComet

Se quiere implementar el algoritmo de segmentación de OpenComet con la finalidad de comparar cuantitativamente sus resultados con los del algoritmo propuesto de segmentación. Se incluirá también en la aplicación como algoritmo opcional. Para realizar este sprint se utilizará la documentación y código fuente de OpenComet.

Sprint 3: Implementación del algoritmo propuesto

En base a las características de las imágenes de las que se dispone, el objetivo de este sprint consiste en desarrollar un algoritmo que segmente de forma eficaz los cometas. Para ello se estudiarán las imágenes en posesión, se implementará un algoritmo de segmentación propio que se adapte a las imágenes y se realizará una comparativa de resultados entre el algoritmo implementado de OpenComet y el nuevo algoritmo.

Sprint 4: Implementación de la aplicación

Se construirá una aplicación con interfaz gráfica que permita al usuario crear, guardar y abrir proyectos, añadir, eliminar y modificar imágenes, segmentar imágenes de forma automática con los algoritmos implementados o manualmente y, por último, modificar y ver los parámetros de los cometas segmentados.

Sprint 5: Redacción de la memoria

Se redactará una memoria en la que se reflejará el trabajo realizado (motivaciones, objetivos, ejecución...) que servirá de documentación y manual de usuario.

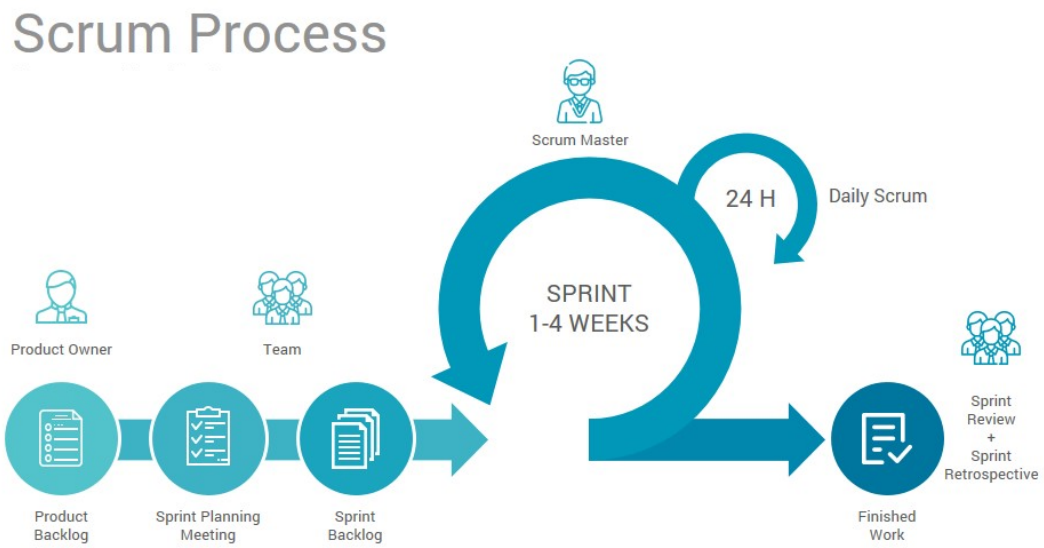


Figura 2.1: Ciclo de vida de la metodología de desarrollo Scrum.

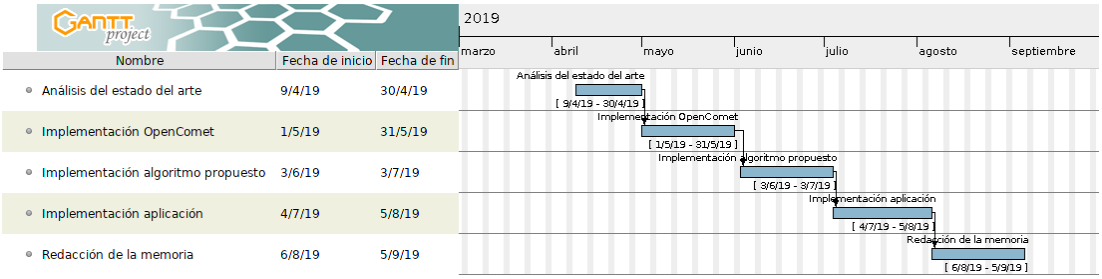


Figura 2.2: Planificación del proyecto. A la izquierda, los sprints originalmente establecidos. A la derecha, el cronograma previsto.

2.3 Seguimiento

El cronograma real con los sprints y el tiempo en que se terminaron se muestra en la imagen de la Figura 2.3. El inicio del primer sprint (*Análisis del estado del arte*) se aplazó hasta la finalización de los estudios y se terminó antes de lo previsto. Esto es así porque el número de aplicaciones de código abierto fue menor del esperado. El segundo sprint (*Implementación de OpenComet*) se terminó según lo planeado. La realización del tercer sprint (*Implementación del algoritmo propuesto)* llevó tres veces más del tiempo previsto. Esto se debe a una reducción en el número de horas diarias dedicadas al sprint. También se debe a que en general el desarrollo y realización de pruebas se alargó más tiempo del esperado. El desarrollo del cuarto sprint (*Implementación de la aplicación*) también se vio retrasado. La razón es que en general, al igual que en el anterior sprint, duró más tiempo del previsto. No obstante, otra razón significativa fue que al verse la entrega del proyecto aplazada para febrero del año siguiente y disponer de holgura suficiente, se invirtió tiempo adicional en funcionalidades no estrictamente primordiales de la aplicación.

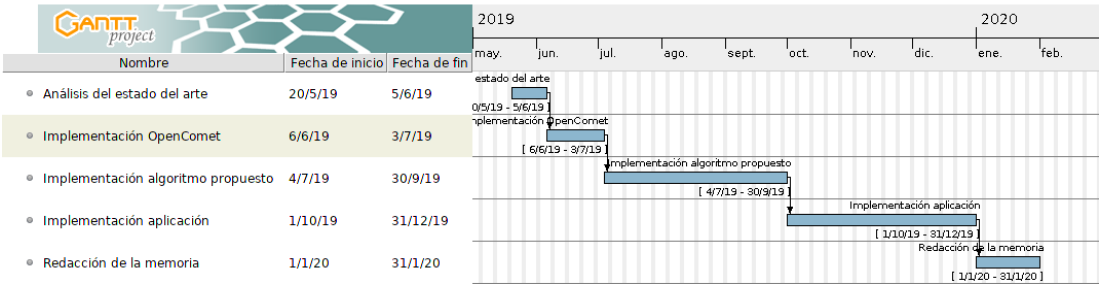


Figura 2.3: Seguimiento del proyecto. A la izquierda, los sprints originalmente establecidos. A la derecha, el cronograma con los tiempos finales.

2.4 Estimación de costes

La estimación de costes se realizó en base a la existencia de dos perfiles en el proyecto: un investigador/programador/analista y un jefe de proyecto. Partiendo de los salarios promedio de un investigador, un programador *junior*, un programador analista y un jefe de proyecto de TI en España [18, 19, 20, 21], se estiman unos salarios de 10.33€/hora y 18.02€/hora respectivamente. Suponiendo que el investigador/programador/analista ha dedicado 4.5 horas de media diarias en el desarrollo del proyecto, los costes estimados para cada sprint se muestran en la Tabla 2.1. Suponiendo que el jefe de proyecto ha invertido 40 horas en el desarrollo del proyecto, sus costes y los costes totales del proyecto se muestran en la Tabla 2.2.

Tabla 2.1: Coste estimado de cada sprint en base al tiempo planificado para el perfil de investigador/programador/analista.

Sprint	**Esfuerzo (h)**	**Coste (€)**
1. Análisis del estado del arte	72	744
2. Implementación de OpenComet	103.5	1069.5
3. Implementación del algoritmo propuesto	103.5	1069.5
4. Implementación de la aplicación	103.5	1069.5
5. Redacción de la memoria	103.5	1069.5

Tabla 2.2: Coste estimado total del proyecto a partir de los costes totales de cada perfil.

Perfil	**Coste (€)**
Investigador/programador/analista	5020.38
Jefe de proyecto	720.8
TOTAL: 5740.46€	

Capítulo 3

Materiales y tecnologías

Para la elaboración del proyecto se tuvo acceso a muestras de células espermáticas de centollo tratadas por el ensayo del cometa. También se ha requerido del uso de distintas herramientas con distintos fines. En el capítulo presente se introducen y se indica qué utilidad han tenido en el proyecto. También se adjunta una tabla con las versiones utilizadas de cada una. Finalmente se habla de las características técnicas de las imágenes del ensayo y cómo se han interpretado.

3.1 Lenguaje de programación y librerías

Python [22] ha sido el lenguaje de programación escogido para este proyecto porque dispone de múltiples librerías propias y de terceros tanto para el procesamiento de imágenes como para la creación de interfaces gráficas. Estas librerías han sido esenciales en la elaboración del algoritmo automático de segmentación y la aplicación. Además, Python es de código abierto, dispone de una buena documentación y de una extensa y activa comunidad. Ambos aspectos han facilitado el diseño y la codificación.

Para representar y manipular las imágenes en el modelo se han utilizado las matrices de la librería *NumPy* [23]. Estas imágenes han sido procesadas con algoritmos de visión artificial obtenidos de la librería *OpenCV* [24]; en menor medida se ha hecho también uso de la librería *scikit-image* [25]. También se utilizó la librería de aprendizaje automático *scikit-learn* [26].

Con respecto al entorno gráfico, se ha utilizado *GTK* [27] para crear los componentes de la interfaz gráfica y configurar su funcionamiento. En ocasiones se ha utilizado *GDK* [28] directamente. También se ha utilizado la librería gráfica *Pycairo* [29]. Para el renderizado de imágenes se ha utilizado la librería de procesamiento de imágenes *GdkPixbuf* [30]. Otra librería de utilidades que se ha utilizado y que va de la mano de GTK y GDK es *GLib* [31].

Para finalizar con el listado de librerías de terceros, la aplicación se ha internacionalizado con la librería de internacionalización *gettext* [32]. Las versiones se adjuntan en la Tabla 3.1.

Tabla 3.1: Tabla con los nombres y versiones de los recursos utilizados para el desarrollo del proyecto.

Nombre	Versión
Python	3.6.9
NumPy	1.18.1
OpenCV	4.1.2.30
scikit-image	0.16.2
scikit-learn	0.22.1
GTK	3.22.30
GDK	3.22.30
GdkPixbuf	2.36.11-2
Pycairo	1.19.0
GLib	2.62.2
gettext	0.19.8.1
Glade	3.22.1
Dia	0.97.2
GanttProject	2.8.10
KolourPaint	17.12.3
Microsoft Paint	1903
Adobe Photoshop Touch	1.4.2
Texmaker	5.0.3

3.2 Herramientas

Para acelerar y facilitar el desarrollo de la interfaz gráfica con GTK se ha empleado *Glade* [33]. Glade es una herramienta que permite fácil y visualmente configurar los componentes. Dicha configuración se almacena en un archivo *XML* al que la aplicación tiene acceso.

Para la creación de los diagramas y de los cronogramas se han empleado *Dia* [34] y *GanttProject* [35], respectivamente. Para la manipulación manual de imágenes se utilizaron los programas *KolourPaint* [36] y *Microsoft Paint* [37]. Para la segmentación manual de los cometas en las imágenes se utilizó *Adobe Photoshop Touch* [38]. Finalmente, para la redacción de este documento se ha utilizado *Texmaker* [39]. Las versiones se adjuntan en la Tabla 3.1.

3.3 Imágenes

Se dispuso de dos conjuntos de imágenes del ensayo del cometa. El primer conjunto está compuesto por un total de 13 imágenes con 75 cometas válidos que se segmentaron manualmente. Cuatro ejemplos de estas imágenes segmentadas manualmente se adjuntan en la Figura 3.1. Este conjunto de imágenes se utilizó para el desarrollo del algoritmo de segmentación y la realización de pruebas. El segundo conjunto consta de 48 imágenes y se utilizaron para

evaluar el daño en su ADN por lo que no se segmentaron manualmente.

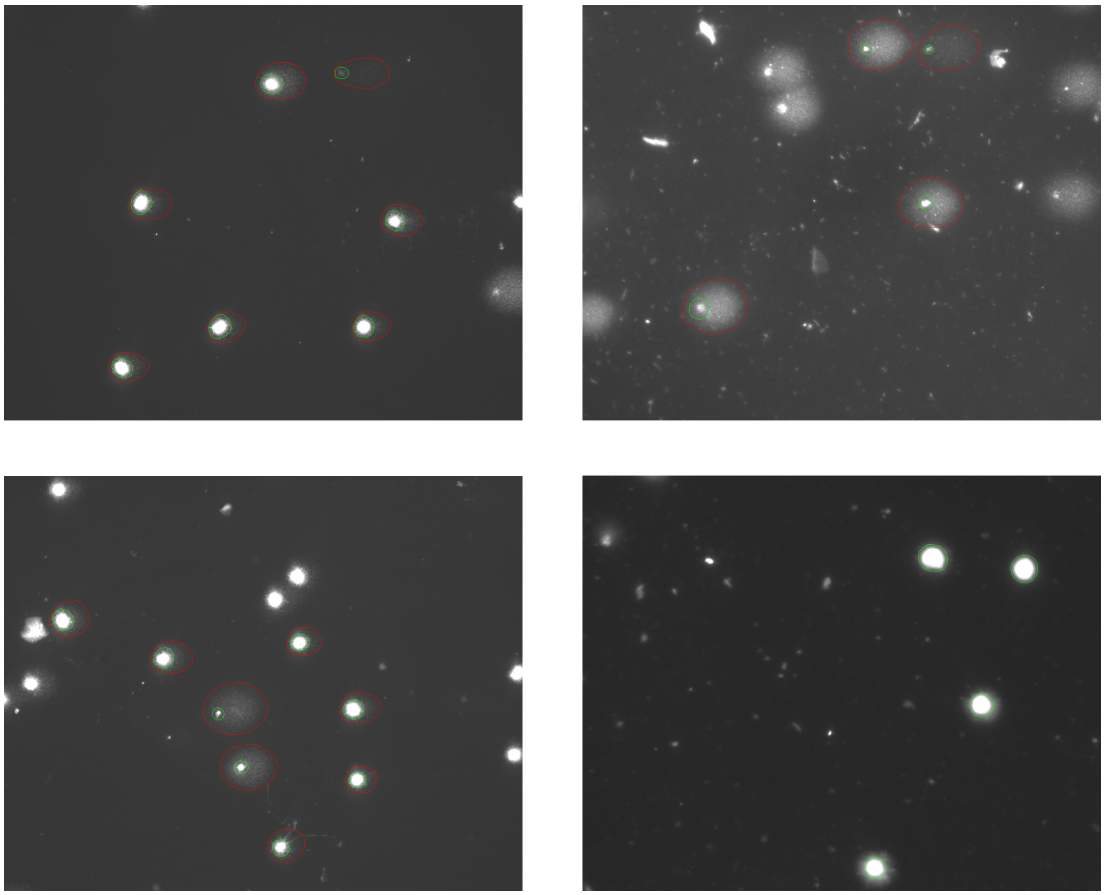


Figura 3.1: Imágenes de células espermáticas de centollo tratadas por el ensayo del cometa y segmentadas manualmente. El contorno rojo delimita la cola del cometa y el verde la cabeza.

Todas las imágenes estaban originalmente en formato *jpeg* y sus dimensiones eran 1280x1024 píxeles. Algunas imágenes tenían los cometas con la cabeza en el lado derecho y la cola en el izquierdo. Estas imágenes fueron volteadas horizontalmente. Esto se debe a que la forma estándar de interpretar los cometas por los programas es con la cabeza en el lado izquierdo del cometa y la cola en el lado derecho.

Capítulo 4

Metodología de segmentación

En este capítulo se desglosa la metodología utilizada en la segmentación automática de las imágenes y se exponen los resultados obtenidos.

4.1 Análisis de histograma

Previo al desarrollo de los algoritmos empleados es conveniente destacar ciertas características de las imágenes a tratar. En ellas coexisten tres tonalidades claramente distinguibles. La primera es una tonalidad muy clara o clara que, por lo general, se asocia a los núcleos o cabezas de los cometas. La segunda tonalidad es una intensidad generalmente intermedia, y se asocia a la cola de los cometas. Finalmente, la tonalidad más oscura pertenece al fondo de la imagen. Esto sucede en términos generales, pues un núcleo suficientemente degradado puede tener prácticamente el mismo nivel de gris en la cabeza que en la cola. Además, las imágenes también pueden presentar ruido u objetos anómalos de todas las tonalidades. En la Figura 4.1 el lector puede apreciar estas diferentes tonalidades. El núcleo 4 es un núcleo sano con una tonalidad muy clara. El resto de núcleos están degradados, teniendo la cabeza —parte delimitada en verde— con una tonalidad muy clara y la cola —parte delimitada en rojo— con una tonalidad intermedia. El resto de la imagen de tono oscuro es el fondo. Como consecuencia, una sencilla y eficaz aproximación es analizar estas distintas tonalidades e intentar separar la imagen por niveles de gris. Es por ello que el análisis de los histogramas de las imágenes puede resultar útil.

Un histograma [11] no es más que la representación gráfica de los valores de intensidad que toman los píxeles en una imagen. En la Figura 4.2 se muestran dos imágenes con sus respectivos histogramas. En el eje X se indica el rango de valores de intensidad que los píxeles pueden tener y en el eje Y la cantidad de píxeles con dicho valor de intensidad. De este modo, dado un histograma H donde $H(90) = 1500$ nos indica que 1500 de los píxeles de la imagen tienen una intensidad 90.

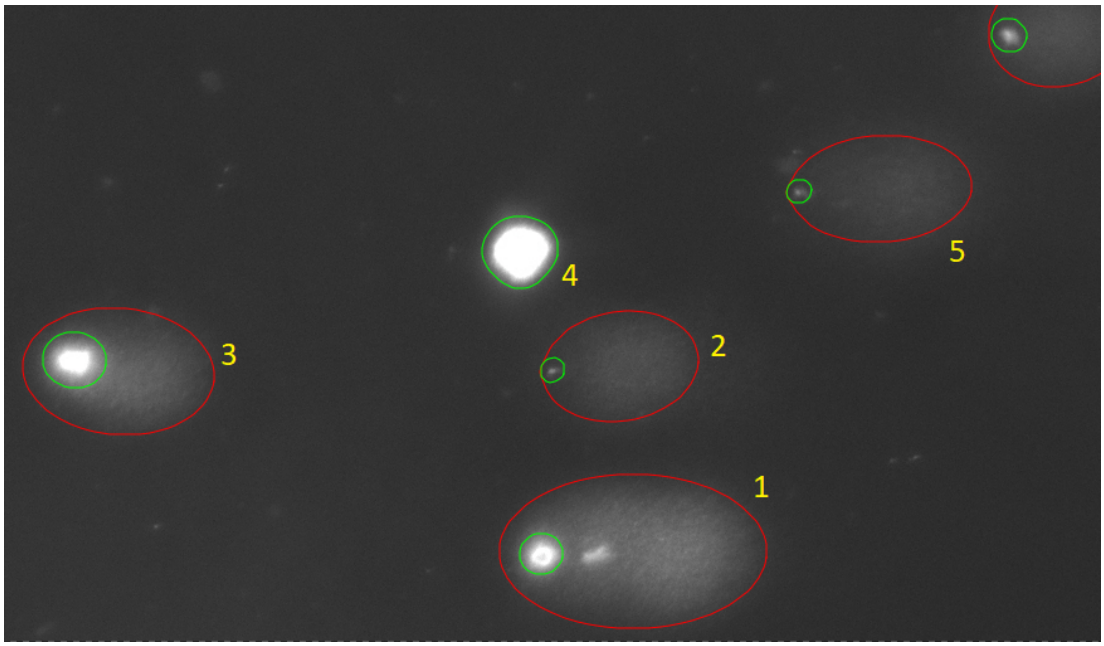


Figura 4.1: Imagen de entrada con cometas segmentados. El contorno rojo es la cola del cometa y el verde la cabeza. Los núcleos 1, 2, 3 y 5 son núcleos degradados con forma de cometa. El núcleo 4 está sano y no tiene cola.

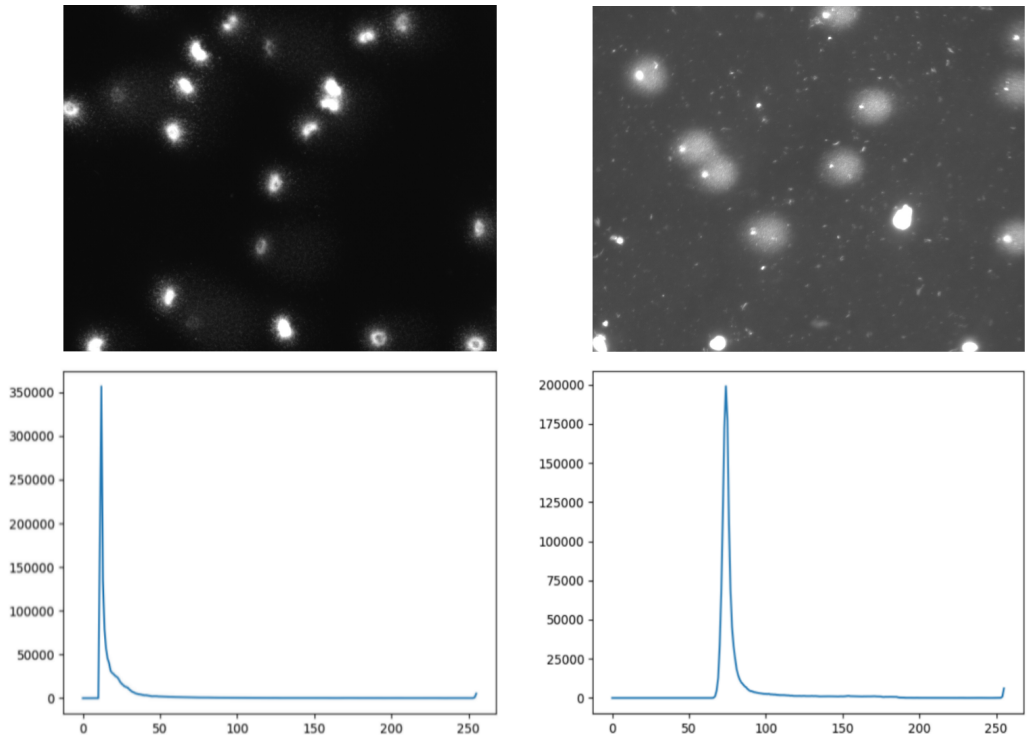


Figura 4.2: Imágenes muestra con sus respectivos histogramas. Izq.: imagen con histograma estándar. Dcha.: imagen con histograma desplazado a la derecha.

En los histogramas de la Figura 4.2 existe un pico que representa el fondo de la imagen. La única diferencia real entre histogramas de imágenes de células cometa es el desplazamiento que pueden presentar, el cual deriva visualmente en una imagen más clara u oscura. Esto depende de si el desplazamiento es a la derecha del eje X, o a la izquierda, respectivamente. Este aspecto puede ser esencial a la hora de separar el fondo de los núcleos en la imagen.

4.2 Identificación de los cometas

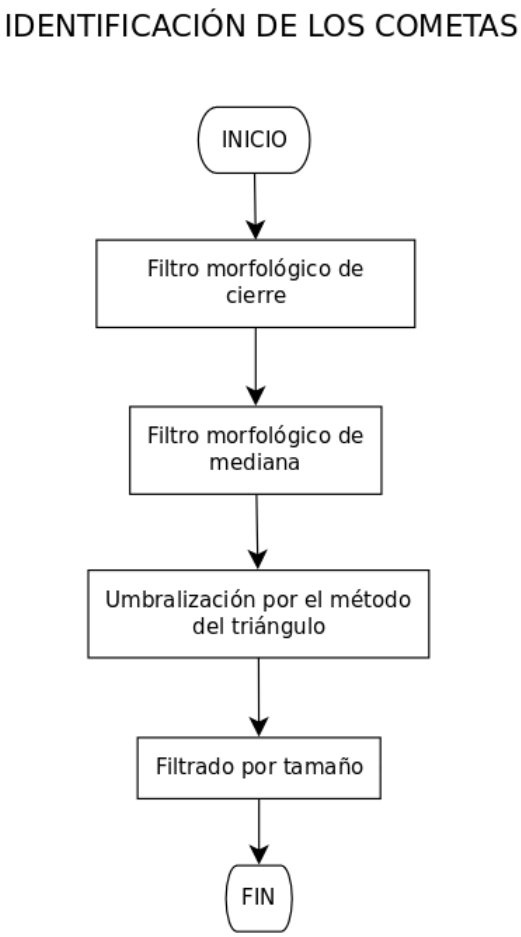


Figura 4.3: Diagrama con los pasos que se siguen en *Identificación de los cometas*.

En esta sección se trata de localizar los cometas que hay en las imágenes. En la Figura 4.3 se resumen los pasos del proceso. Inicialmente, las imágenes de entrada se someten a dos técnicas de filtrado morfológico. La primera técnica que se emplea es el filtro de cierre a nivel de gris [11]. La razón principal es la granularidad que suelen tener las colas de los cometas en estas imágenes. Al aplicar el cierre se obtienen figuras más compactas.

La segunda técnica que se emplea es el filtro de mediana [11], también a nivel de gris. Esto es así porque las imágenes presentan ruido de tipo impulsional o *sal y pimienta* [11]. El filtro de mediana consigue eliminar gran parte de este ruido y suaviza la imagen. Otro aspecto a tener en cuenta es que el elemento estructurante que se utiliza es circular con el objetivo de resaltar la circularidad de los núcleos. En la Figura 4.4 se muestra el resultado de aplicar estos dos filtros con un elemento estructurante circular de radio 5 a una sección de las imágenes de entrada.

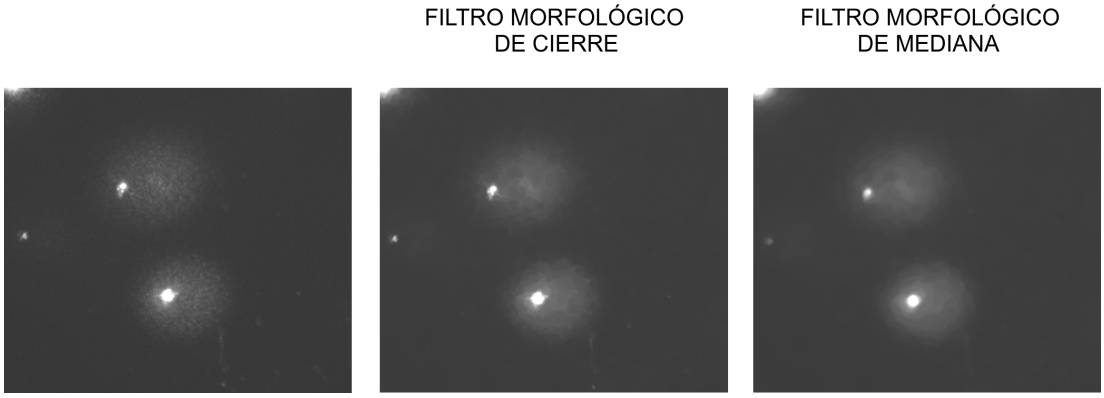


Figura 4.4: Filtro morfológico de cierre seguido de filtro morfológico de mediana con elemento estructurante circular de radio 5.

Una vez preprocesada la imagen, el siguiente paso consiste en separar el fondo de los potenciales cometas. Dadas las características de los histogramas de las imágenes se ha optado por utilizar el método de umbralización [11].

La umbralización es un proceso cuyo objetivo radica en separar los píxeles de una imagen en clases. Por lo general, se pretende dividir la imagen en dos clases: *fondo* y *objetos*, aunque existen algoritmos que logran dividir la imagen en k clases de forma eficaz. Esto se consigue en base a los valores de intensidad de los píxeles e incluso en base a la forma del histograma. El algoritmo determina qué valor de intensidad T (o valores, si se opta por dividir en más de dos clases), conocido como umbral, divide de forma óptima la imagen. Una vez establecido el umbral T, el proceso para dividir una imagen X de $M \times N$ dimensiones en una imagen de dos clases Y viene definido por la siguiente ecuación:

$$Y(i,j) = \begin{cases} maxValue, & \text{if } X(i,j) \geq T \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

donde (i,j) representa el píxel de la imagen situado en la fila i y columna j, siendo $1 \leq i \leq M$ y $1 \leq j \leq N$.

Son tres los métodos de umbralización que se han probado: el método del *triángulo* [40],

el método de Huang [41] y el método de Otsu [42].

El método del *triángulo* trabaja directamente sobre la forma del histograma. Su cómputo construye una línea entre el valor máximo de la función y el último punto del eje *X*. El algoritmo determina el umbral buscando la línea perpendicular de longitud máxima entre la hipotenusa del triángulo que se forma y la gráfica del histograma, tal y como se puede ver gráficamente en la Figura 4.6.

El segundo algoritmo es el método de Huang y se basa en aplicar la teoría de los conjuntos difusos para minimizar la entropía de la imagen.

El tercer y último algoritmo es el algoritmo de Otsu, el cual determina el umbral óptimo minimizando la varianza dentro de las clases o lo que es lo mismo, maximizando la varianza entre clases.

En la Figura 4.5 se muestran los resultados de aplicar los tres algoritmos de umbralización binaria sobre una misma imagen. En la fila inferior se organizan las imágenes binarias, resultado de aplicar la ecuación 4.1 sobre las imágenes originales con el umbral obtenido tras ejecutar el método de umbralización correspondiente. Así pues, los píxeles de color negro $Y(i,j) = 0$ pertenecen a la clase *fondo*, y los píxeles blancos $Y(i,j) = maxValue$ pertenecen a la clase *objetos*. El lector puede apreciar que el conjunto de píxeles de la clase *objetos* es cada vez más pequeño de izquierda a derecha. Esto se debe a que el valor umbral de intensidad obtenido es mayor.

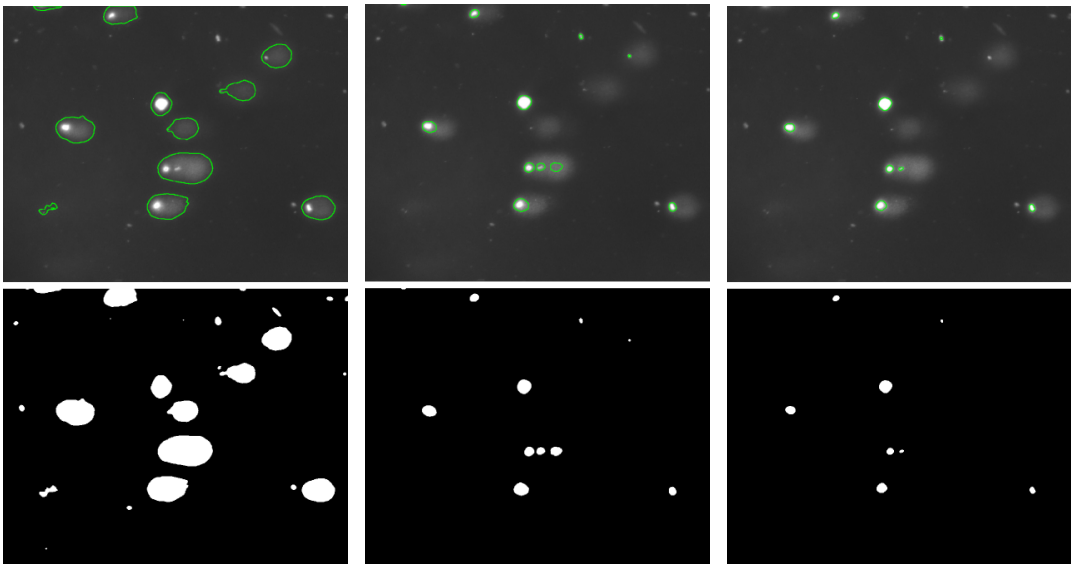
El algoritmo de umbralización que se eligió aquí fue el del *triángulo* porque como se ve en la Figura 4.5 los contornos resultantes abarcan una mayor área de los cometas.

A raíz de las imágenes binarias obtenidas tras la umbralización, se sustraen todas las regiones conexas del conjunto *objetos*. Para cada región se calcula su área y se descarta si su tamaño no supera un umbral τ, el cual define el tamaño mínimo que una región ha de tener para seguir siendo procesada. En la Figura 4.7 se muestra el resultado de aplicar el filtrado por tamaño en una de las imágenes binarias que se obtiene tras la primera umbralización.

Se asume que las imágenes están tomadas siempre desde aproximadamente la misma distancia, por lo que el tamaño de los núcleos en proporción a la imagen no debería variar significativamente de una imagen a otra. Descartar estos contornos que a priori se sabe que no son núcleos reduce cómputo innecesario futuro.

4.3 Segmentación de la cabeza

Cada región conexa obtenida en el paso anterior es un potencial cometa y en él se intenta localizar su cabeza, la cual es, como norma general, la región más clara. Para este procedimiento se trabaja sobre la imagen original en escala de grises. De este modo contamos con los niveles de intensidad originales y no se pierde información. En el diagrama de la Figura 4.8 se



(a) (b) (c)

Figura 4.5: Distintos algoritmos de umbralización. (a) Método del *triángulo*. (b) Método de Huang. (c) Método de Otsu. En la fila superior; contornos de las imágenes binarias dibujados sobre la imagen original. En la fila inferior; las imágenes binarias, resultado de aplicar el umbral obtenido tras ejecutar el algoritmo sobre la imagen original.

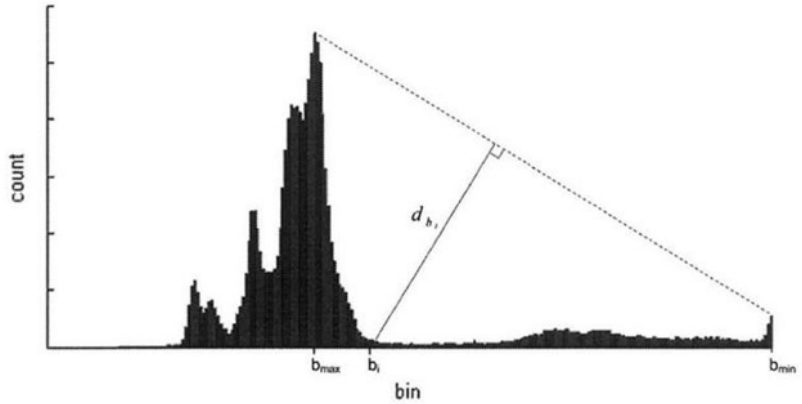


Figura 4.6: Representación gráfica de aplicar el método del *triángulo* sobre un histograma. b_{max} es el punto donde el histograma alcanza el valor máximo y b_{min} el último punto del eje X. d_{h_i} es la recta perpendicular de longitud máxima y b_i el punto en el eje X que se corresponde con el valor de intensidad que el algoritmo selecciona como umbral por defecto.

resume el proceso de esta sección.

La segmentación de la cabeza de los cometas se divide en dos fases: *Segmentación inicial* y *Validación de la segmentación.*

4.3.1 Segmentación inicial

No es realista esperar que los cometas se presenten siempre impolutos (Fig. 4.9a). En los cometas hay ruido (Fig. 4.9b), el cual puede dificultar la localización de la cabeza. La cabeza puede estar lo suficientemente degradada de forma que, a pesar de que forme visualmente una figura circular u ovalada, esté constituida por más de una tonalidad (Fig. 4.9c). En otras ocasiones no está ni del todo claro cuál sería la correcta segmentación (Fig. 4.9d). Estos escenarios dificultan su debida segmentación. Por esta razón se procesan previamente las regiones de los cometas para resaltar el área circular de la cabeza e intentar eliminar el ruido. Así, la segmentación y salida visual ofrecida al usuario pueden ser coherentes con la naturaleza de los núcleos de las células.

Para esto se emplean dos técnicas de filtrado morfológico: el filtro de dilatación [11] y el filtro de mediana. Cada una de las regiones es procesada con el filtro de dilatación seguido del filtro de mediana. Para ambos se emplea un elemento estructurante circular. Este proceso se repite dos veces. La dilatación expande los perímetros de la cabeza e iguala los niveles de intensidad con el fin de obtener una cabeza circular sin huecos y de una misma tonalidad. El filtro de mediana elimina el ruido y suaviza la imagen. En la Figura 4.10 hay dos ejemplos de dos cometas de distintas imágenes de entrada sometiéndose a estos filtros. Obsérvese como se cumplen las premisas descritas.

Puesto que los niveles de intensidad en la cabeza del cometa suelen ser más altos con respecto al resto de la región, la primera fase concluye con el empleo del método de umbralización de Otsu para intentar separar la región de la cabeza de la cola. En la Figura 4.11 están las regiones procesadas de la figura anterior junto con los resultados de aplicar el método de Otsu. Obsérvese que los contornos delimitan correctamente la cabeza del cometa. Sin embargo, pueden surgir dos situaciones desfavorables que requieren de mayor procesamiento.

La primera ocurre cuando después de umbralizar son más de una las regiones conexas que pertenecen al conjunto *objetos* (Fig. 4.12, imagen superior). Es decir, en el cometa hay varias regiones con niveles elevados y similares de intensidad. Esto puede ocurrir si en la región del cometa hay ruido o si en realidad son más de uno los cometas que coexisten en la misma región. Una sencilla aproximación sería descartar por completo estos cometas. En la metodología propuesta se intenta corregir y se mantienen para más adelante decidir qué candidato como cabeza del cometa es mejor.

El segundo problema se produce cuando la umbralización resulta en una única región conexa pero el contorno no se corresponde con el perímetro real de la cabeza (Fig. 4.12, imagen

inferior).

4.3.2 Validación de la segmentación

Esta fase consiste en realizar sucesivas umbralizaciones con Otsu sobre las cabezas candidatas de los cometas obtenidas en *Segmentación inicial*. El número de veces que son umbralizadas viene determinado por la validez de su segmentación, hasta un número máximo ϵ de umbralizaciones. Los criterios que definen la validez de la segmentación son el tamaño del candidato, la convexidad, la circularidad y la proporción entre el tamaño de este con respecto al tamaño total del cometa.

Un candidato de pequeño tamaño no es conveniente umbralizarlo porque las cabezas pequeñas suelen ser núcleos muy degradados. En estos núcleos muy degradados sólo unos pocos píxeles tienen valores altos de intensidad. Una continua umbralización derivaría en un perímetro muy pobre e ineficiente. Por otro lado, las cabezas son convexas y circulares. Si una región no lo es, podría tratarse de una cabeza válida que su primera umbralización resultara en una segmentación que no hiciera justicia a su perímetro real. Finalmente, los cometas, cuanto menos degradados, mayor el tamaño de su cabeza, menor el tamaño de la cola y por ende mayor la proporción entre el tamaño de la cabeza y el tamaño del cometa. Como en pasos anteriores se ha agrandado el tamaño de la cabeza, sucesivas umbralizaciones pueden lograr que el contorno resultante se ajuste más al real.

En la Figura 4.13 se muestran dos ejemplos de sucesivas umbralizaciones. La umbralización en la cabeza de la fila superior es necesaria porque la circularidad es demasiado baja. En la fila inferior, la umbralización surge a raíz del valor de la variable *proporción*.

4.3.3 Validación de la cabeza

Después de obtener las regiones candidatas a cabeza y haber validado su segmentación, el siguiente paso consiste en decidir qué región será la cabeza del cometa. Esto es así porque un cometa sólo puede tener una cabeza.

Antes de comenzar las cabezas son validadas. Las cabezas que no son válidas dejan de ser candidatas. Son cuatro los criterios que se tienen en cuenta a la hora de validar. El primer factor y más decisivo es si el contorno de la cabeza está tocando o no los bordes de la imagen. Las cabezas que están tocando algún borde de la imagen no son válidas. Esto es así porque el análisis de estas cabezas no sería válido al no poder analizar la totalidad del cometa. Otro factor es el tamaño. Las cabezas que no superan un tamaño τ mínimo no se consideran válidas. Esto es así porque lo más probable es que se trate de ruido. Las dos últimas características son la convexidad y la circularidad. Los núcleos son de naturaleza convexa y circular. Una región del cometa no puede ser candidata a cabeza si no supera unos niveles mínimos de convexidad y circularidad. Especialmente después de haber sido procesada con filtros afines y elementos

estructurantes circulares. En la Figura 4.14 se adjunta un posible escenario. En la imagen de la izquierda, los contornos de color rojo delimitan los cometas. En su interior y en verde se hallan los contornos de las cabezas candidatas. En la imagen de la derecha, las regiones de color verde son cabezas válidas para seguir siendo candidatas. Las regiones de color rojo no son válidas y se descartan. En concreto, las regiones rojas 1, 2 y 3 tocan el borde inferior de la imagen por lo que no pueden ser candidatas. La región 4 se descarta por su tamaño al no superar el tamaño mínimo. La región 5 toca el borde superior y tampoco cumple con los requisitos de convexidad y circularidad.

Si un cometa se queda sin candidatos válidos es descartado. Si el cometa tiene únicamente un candidato está listo para pasar a la siguiente fase del algoritmo. Para los cometas que continúan teniendo más de un candidato se decide por el candidato óptimo. La elección del candidato óptimo se realiza en base a que la cabeza del cometa debería estar situada a la izquierda de este a una altura central. Así, se define un punto pivote p en este espacio para cada cometa. La región candidata que esté más próxima a ese punto pasa a ser la cabeza del cometa. La distancia al punto pivote se mide desde el centroide de los candidatos y se calcula como distancia euclídea. Siendo $p_1 = (p_{1_x}, p_{1_y})$ y $p_2 = (p_{2_x}, p_{2_y})$ dos puntos en el plano bidimensional, su distancia euclídea $d(p_1, p_2)$ se computa como:

$$d(p_1, p_2) = \sqrt{(p_{1_x} - p_{2_x})^2 + (p_{1_y} - p_{2_y})^2} \tag{4.2}$$

Este procedimiento se puede ver de forma más representativa en la Figura 4.15. La región verde en las imágenes de la derecha representa la región elegida como cabeza en cada cometa. Esto es así porque en la imagen superior $d(p, p_1) < d(p, p_2) < d(p, p_3)$ y en la imagen inferior $d(p', p'_1) < d(p', p'_2)$, siendo p, p' los puntos pivote y $p_1, p_2, p_3, p'_1, p'_2$ los centroides.

4.4 Segmentación de la cola

Los contornos de los cometas son irregulares debido a objetos extraños del fondo y el ruido (Fig. 4.16). Los cometas deberían tener una forma circular u elíptica sin rugosidades. Por esta razón se trata de determinar qué contorno define mejor la cola de los cometas. El proceso que se sigue en esta sección se puede ver en la Figura 4.17.

Para intentar paliar la existencia de objetos anómalos alrededor de los cometas se usa en una primera instancia el filtro morfológico de apertura [11]. El elemento estructurante utilizado es de forma elíptica con la elipse orientada horizontalmente. Esta técnica elimina las irregularidades en los bordes. A posteriori se obtiene la envolvente convexa [11] del contorno de la máscara resultante, que pasará a ser el nuevo contorno sin rugosidades del cometa.

Algunos cometas no están degradados y no deberían tener cola. Por esta razón, después de haber corregido el contorno de los cometas es necesario decidir si el cometa debería tener

o no cola. Para esta tarea se ha entrenado un árbol de decisión [43].

El árbol de decisión predice en base a cuatro parámetros. El primer parámetro es la proporción entre el tamaño de la cabeza y el tamaño del cometa (eq. 4.3). A mayor valor en la proporción más probable que se trate de un cometa sano. El segundo y tercer parámetros son el valor promedio de intensidad de los píxeles que pertenecen a la cabeza (eq. 4.4) y de los píxeles que pertenecen a la cola (eq. 4.5). Cuanto menos degradado un cometa mayor es la intensidad promedia en la cabeza y menor en la región detectada como cola. En estas ecuaciones, M_{head} representa la máscara binaria con los píxeles del área de la cabeza y M_{tail} la máscara binaria con los píxeles del área de la cola. m y n son las dimensiones de la imagen. El último parámetro se muestra en la ecuación 4.6 y es la diferencia entre la distancia del punto más a la derecha de la cabeza con el punto más a la derecha de la cola y la distancia del punto más a la izquierda de la cola con el punto más a la izquierda de la cabeza. C_{head} representa los puntos del contorno de la cabeza y C_{tail} los puntos del contorno de la cola mientras que p_r y p_l son los puntos del contorno situados más a la derecha e izquierda, respectivamente.

$$\rho = \frac{A_{head}}{A_{comet}} \tag{4.3}$$

$$\overline{I}_{head} = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} I(i,j)M_{head}(i,j)}{\sum_{i=0}^{m}\sum_{j=0}^{n} M_{head}(i,j)} \tag{4.4}$$

$$\overline{I}_{tail} = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} I(i,j)M_{tail}(i,j)}{\sum_{i=0}^{m}\sum_{j=0}^{n} M_{tail}(i,j)} \tag{4.5}$$

$$\begin{aligned} \delta = (C_{head}[p_r](x) - C_{tail}[p_l](x)) - (C_{tail}[p_r](x) - C_{head}[p_l](x)) \\ C_A[p_r](x) > C_A[q](x), \forall q \in C_A, A \subset \{head, tail\} \\ C_A[p_l](x) < C_A[q](x), \forall q \in C_A, A \subset \{head, tail\} \end{aligned} \tag{4.6}$$

Para concluir se validan los cometas según su anchura y altura. Así, los cometas cuya relación entre altura y anchura supera un valor τ son descartados. Esto se debe a que los cometas suelen tener más anchura que altura. De este modo se eliminan gran parte de los cometas solapados.

4.5 Ajustes opcionales

Por defecto, el algoritmo no procesa más los cometas (Fig. 4.18a). No obstante, si así lo ha indicado el usuario, el algoritmo sustituye el contorno que segmenta la cola por la elipse que mejor se ajusta a sus puntos (Fig. 4.18b), el de la cabeza (Fig. 4.18c), o ambos (Fig. 4.18d).

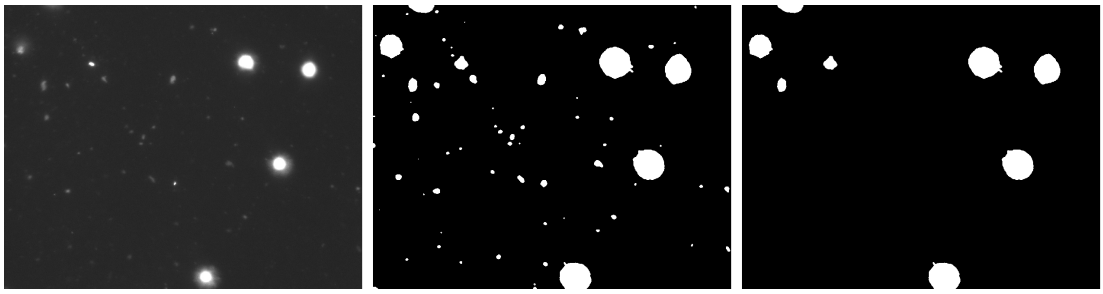


Figura 4.7: Tras la primera umbralización, los contornos que no superan un tamaño τ son descartados. La primera imagen es la imagen original. La segunda imagen es el resultado tras la primera umbralización. La tercera imagen es el resultado de eliminar las regiones conexas que no superan un tamaño umbral τ.

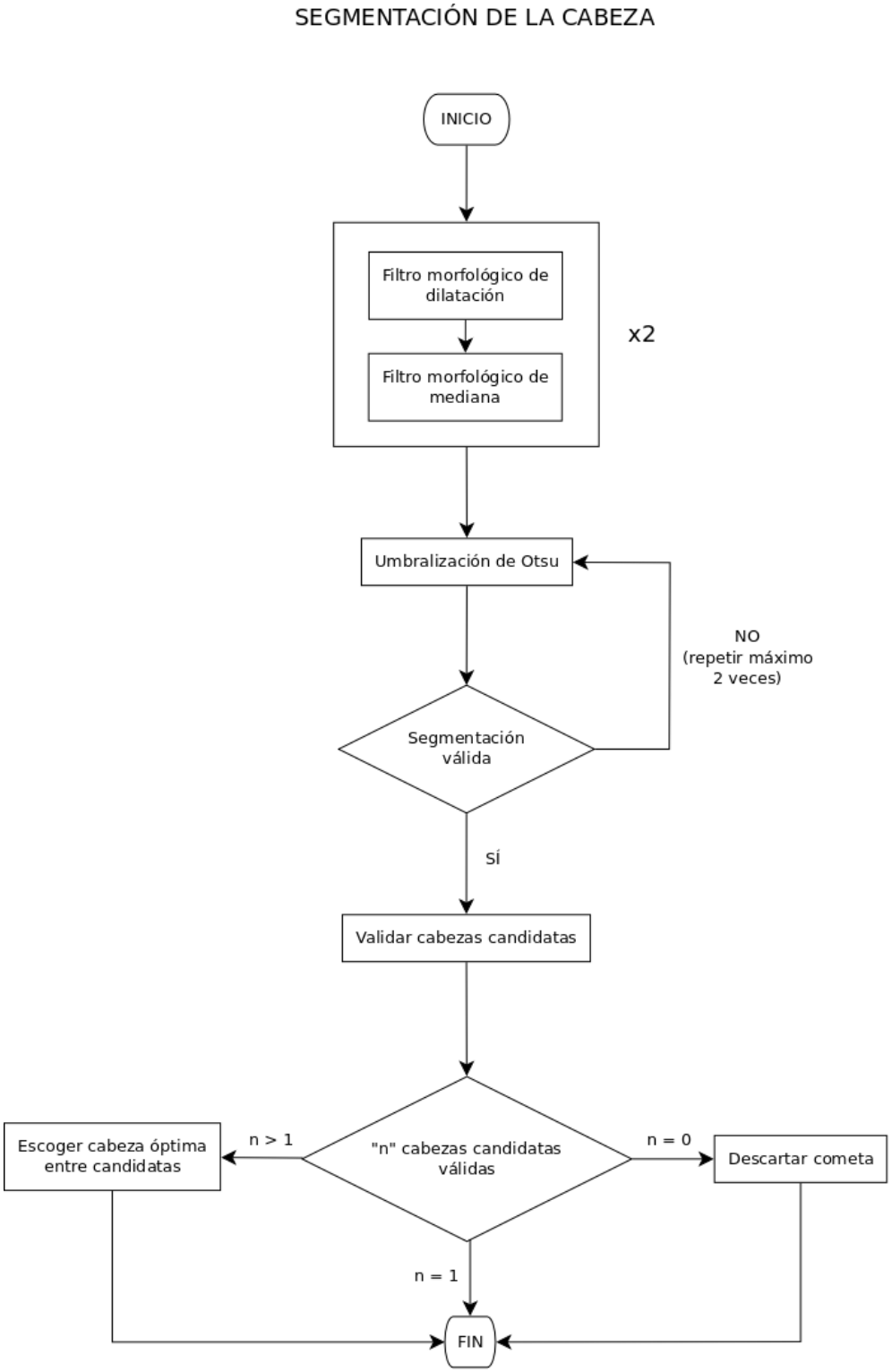
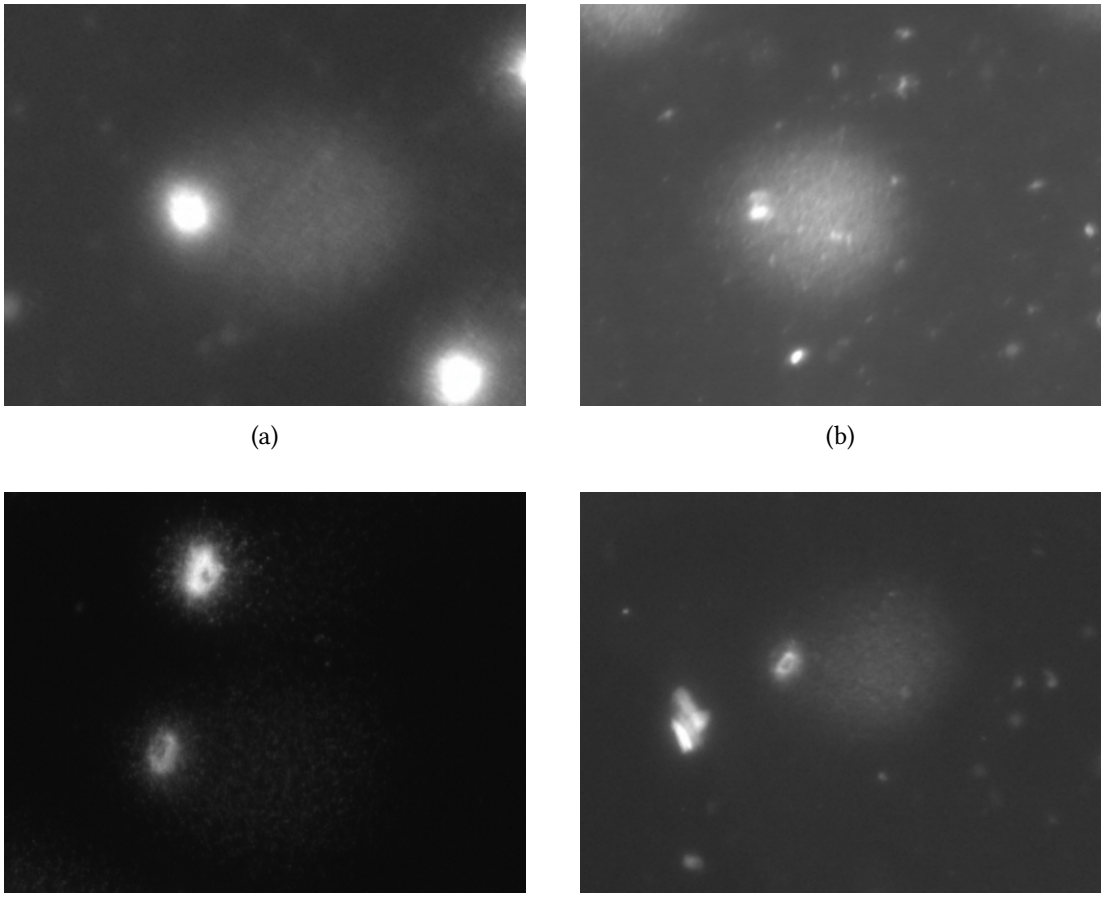


Figura 4.8: Diagrama con los pasos que se siguen en *Segmentación de la cabeza.*



(a) (b)

(c) (d)

Figura 4.9: a) Cometa con sus partes muy bien definidas y distinguibles. b) Cometa con ruido en la cola y cabeza. c) Cometa con la cabeza parcialmente deteriorada. d) Cometa con la cabeza muy deteriorada.

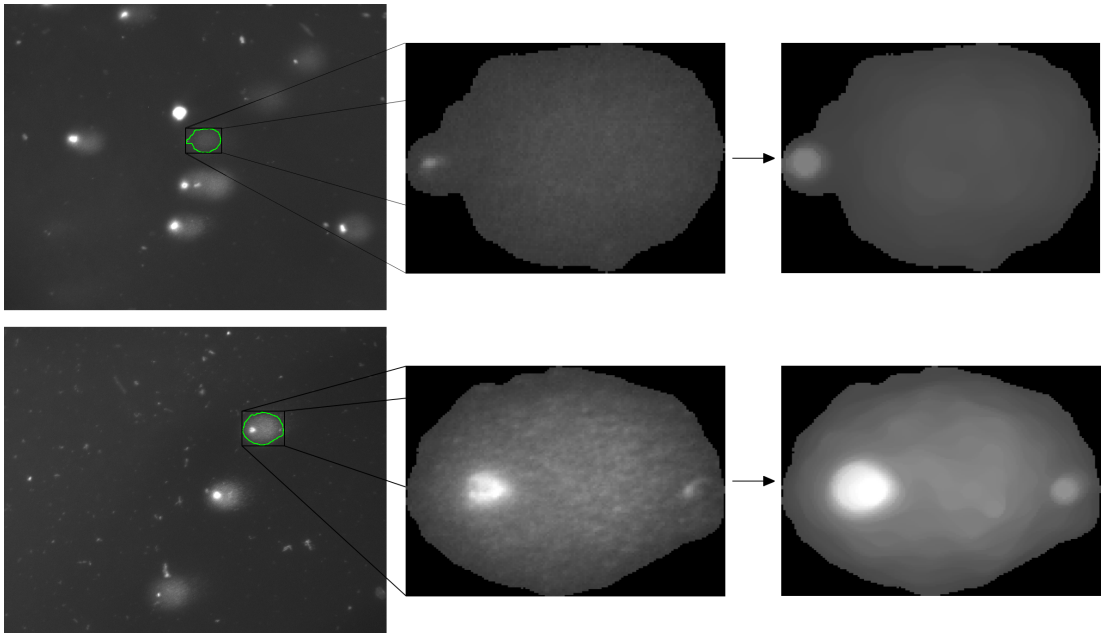


Figura 4.10: Procesado de las regiones de los cometas. A la izquierda, las imágenes originales con la región del cometa que se va a procesar; en el centro, la región del cometa con los niveles de intensidad original; a la derecha, la misma región después de ser procesada con los filtros morfológicos de dilatación y mediana.

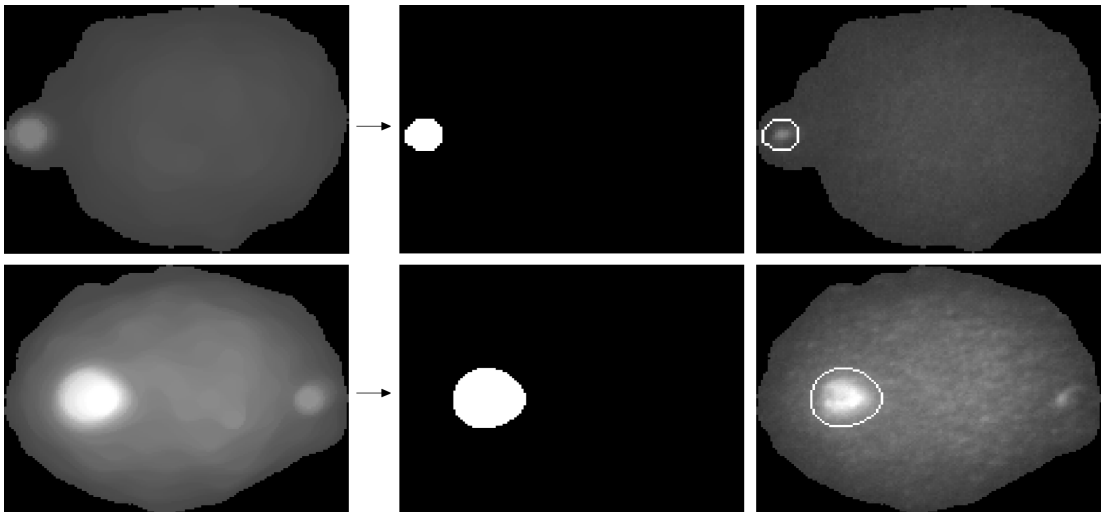


Figura 4.11: Umbralización de la región procesada del cometa por el método de Otsu. A la izquierda, las regiones procesadas de los cometas; en el centro, las imágenes binarias resultantes de aplicar el método de Otsu; a la derecha, las regiones originales de los cometas con los contornos de las imágenes binarias correspondientes dibujados en blanco.

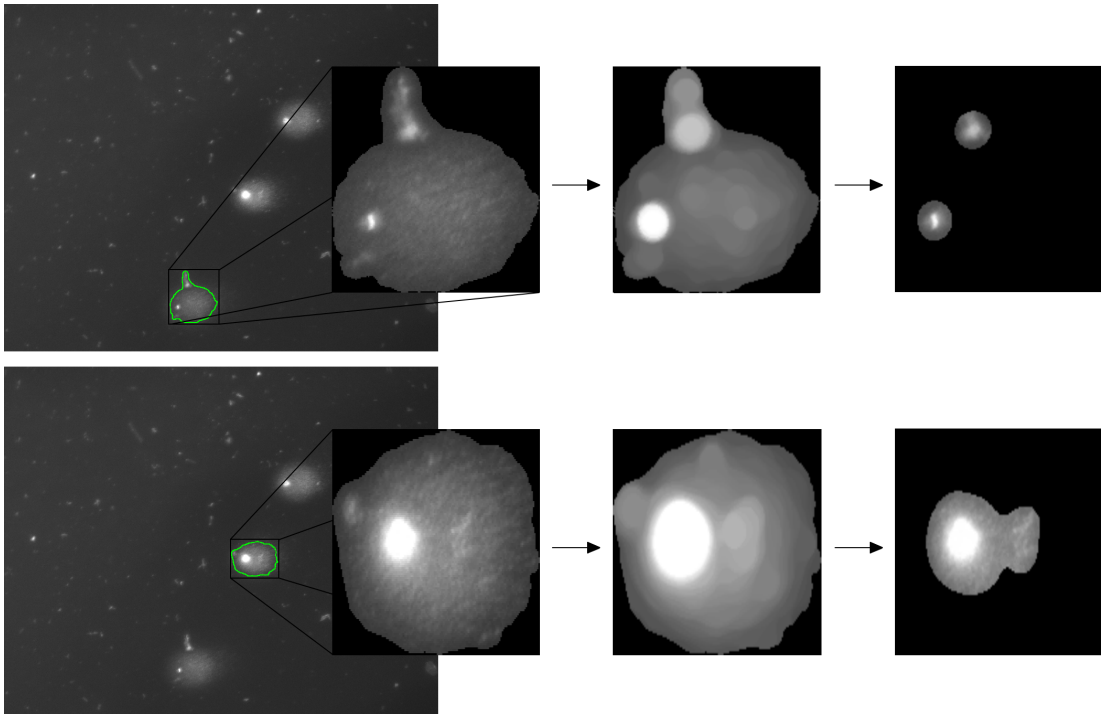


Figura 4.12: El conjunto de imágenes superior muestra cómo la extracción de la región de la cabeza por umbralización resulta en dos regiones. El conjunto de imágenes inferior ejemplifica cuando la región resultante tras la umbralización no delimita correctamente el perímetro real de la cabeza del cometa.

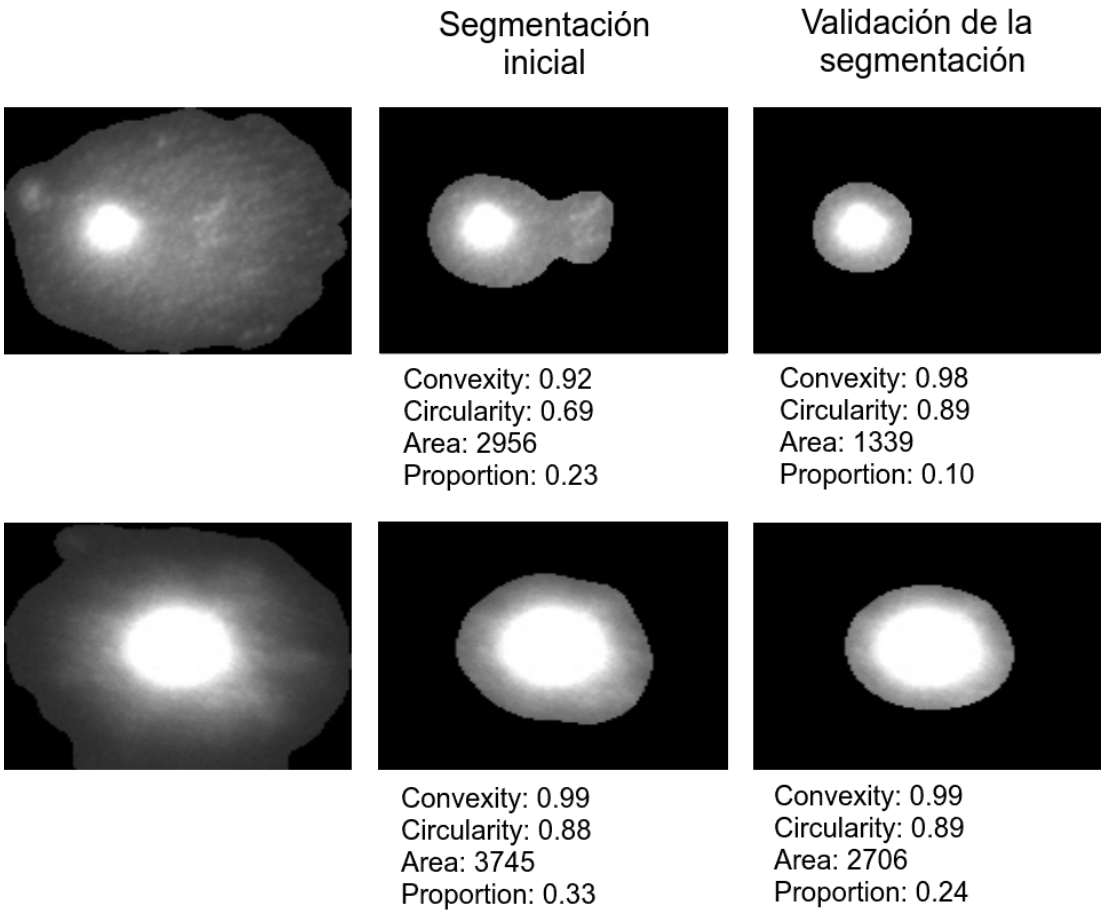


Figura 4.13: Umbralizaciones posteriores en base a la convexidad del contorno de la cabeza, la circularidad, el tamaño y la proporción entre el tamaño del contorno de la cabeza y el contorno del cometa. La primera umbralización se debe a la baja circularidad del contorno. La segunda umbralización surge a raíz del valor de proporción entre el tamaño de la cabeza y el tamaño del cometa.

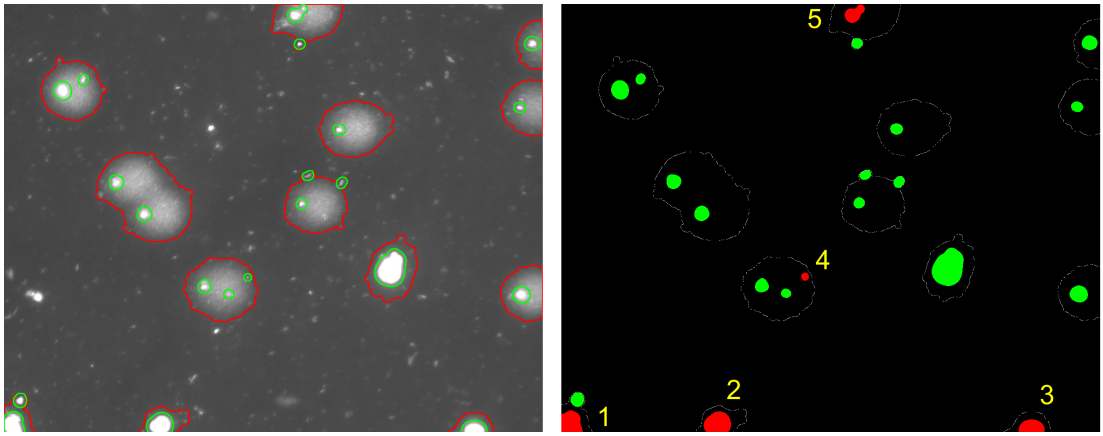


Figura 4.14: A la izquierda, los cometas en rojo y sus respectivas regiones candidatas a cabeza en verde. A la derecha, el resultado de validar las regiones candidatas a cabeza. Las regiones verdes son cabezas válidas; las rojas, no válidas.

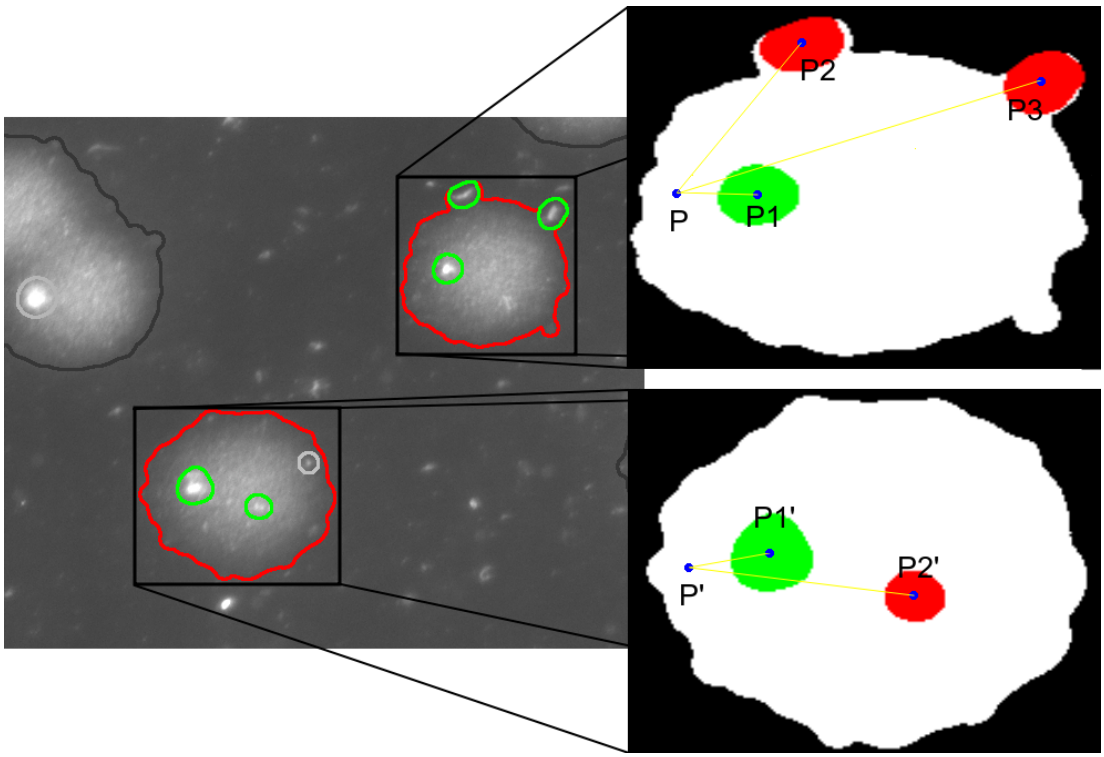


Figura 4.15: La selección del candidato óptimo se realiza en base a la distancia euclídea de los centroides de los candidatos al punto pivote. En la imagen superior se selecciona la región verde como cabeza porque la distancia euclídea de p_1 al punto pivote p es la más pequeña. En la imagen inferior se selecciona la región verde como cabeza porque la distancia euclídea de p_1' al punto pivote p' es la más pequeña.

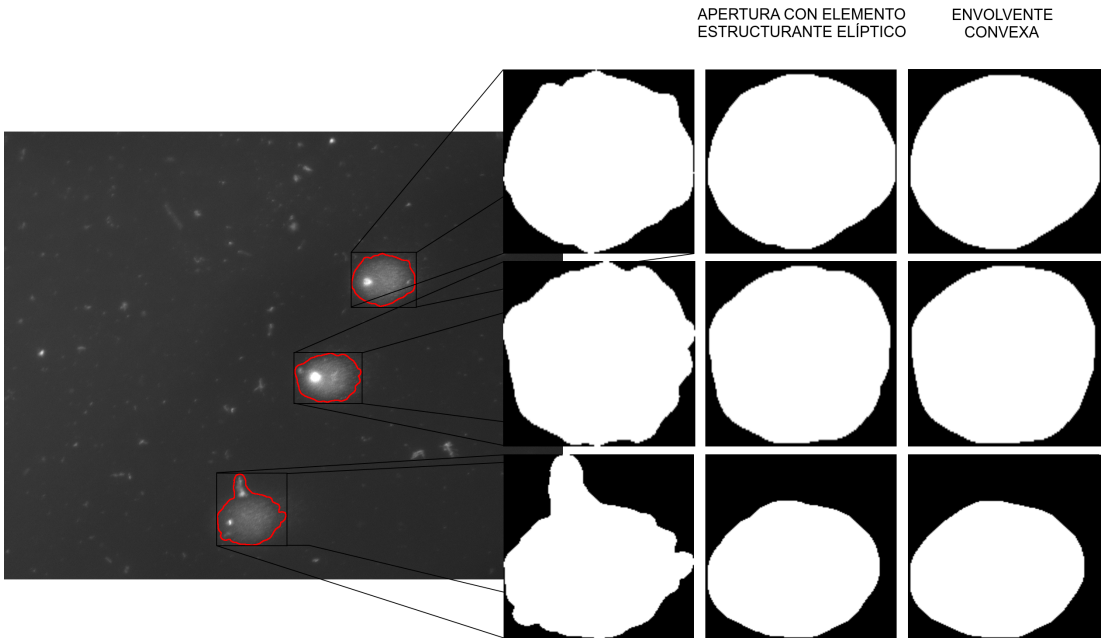


Figura 4.16: Mejora del contorno de los cometas. A las máscaras binarias de los cometas se les aplica un filtro morfológico de apertura con un elemento estructurante elíptico. De la máscara resultante se obtiene la envolvente convexa.

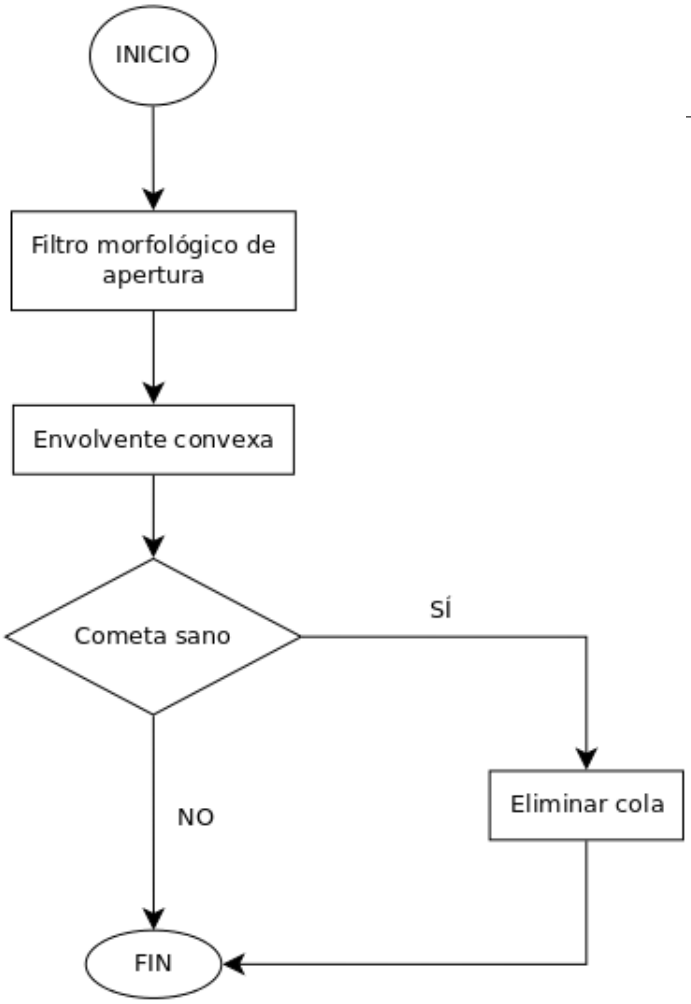


Figura 4.17: Diagrama con los pasos que se sigue en *Segmentación de la cola.*

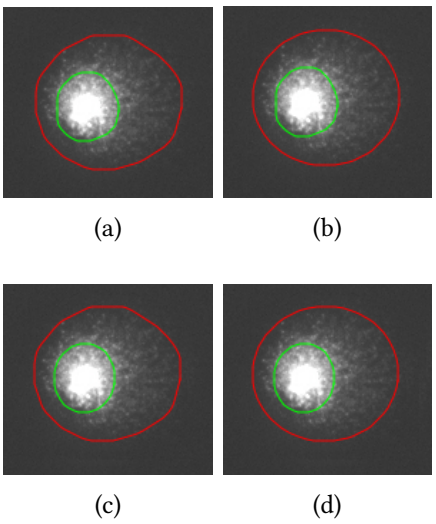


Figura 4.18: a) Cometa con contornos naturales. b) Cometa con contorno de la cola elíptico. c) Cometa con contorno de la cabeza elíptico. d) Cometa con contornos de la cola y cabeza con formas de elipse.

4.6 Resultados

Para evaluar cuantitativamente los resultados de los algoritmos de segmentación se utilizó el conjunto de 13 imágenes que fue segmentado de forma manual. Así, las segmentaciones realizadas a mano se compararon con las creadas por las herramientas automáticas. Cuanto más se aproxime una a la otra, mejor el resultado del algoritmo.

De este modo se emplearon los parámetros que se suelen representar en las matrices de confusión [44] y que determinan cuán efectivo es un modelo de decisión binario. Estos parámetros son *verdaderos positivos* (VP), *falsos positivos* (FP), *verdaderos negativos* (VN) y *falsos negativos* (FN). En la imagen de la Figura 4.20 se ejemplifica a qué representa cada grupo. En la primera columna se dibujan los contornos de las cabezas y en la segunda los de las colas. Las imágenes de la primera fila tienen los contornos que se espera que sean segmentados por la herramienta. En la fila del centro están los contornos que el algoritmo realmente segmenta. En la última fila se dibujan las áreas de los contornos de ambas imágenes y se colorean en base al grupo que pertenecen. Los píxeles de color verde son los que se detectan como parte del objeto cuando realmente son parte del objeto. Pertenecen al grupo de los verdaderos positivos. Los píxeles de color azul se detectaron como parte del objeto sin serlo y pertenecen al grupo de falsos positivos. Los píxeles de color rojo son los que no fueron detectados como parte del objeto y debían haberlo sido. Pertenecen al grupo de falsos negativos. El resto de los píxeles de la imagen que mantiene su color original pertenece al grupo de verdaderos negativos y se entienden como los píxeles que no debían ser detectados como parte del objeto y así se cumplió.

Con los valores de estos parámetros se empleó la métrica *IoU* (*Intersection over Union*). Siendo $A_{detected}$ el área detectada para un objeto y A_{real} el área real de dicho objeto:

$$IoU = \frac{A_{detected} \cap A_{real}}{A_{detected} \cup A_{real}} \tag{4.7}$$

El cómputo se realizó por separado sobre la cabeza y cola de cada cometa. Para cada imagen se obtuvieron los valores promedios de sus cometas y finalmente se calculó la media entre los valores de las 13 imágenes. En la Tabla 4.2 se adjuntan los resultados obtenidos mediante el uso de OpenComet y el algoritmo de segmentación propuesto. El algoritmo propuesto funciona claramente mejor con las imágenes de las que se dispone por lo que su empleo aumenta significativamente las probabilidades de lograr una mejor segmentación. En la Tabla 4.1 se muestran los parámetros que utilizó este algoritmo y en la Figura 4.21 se adjuntan imágenes con varios cometas segmentados por OpenComet y FreeComet.

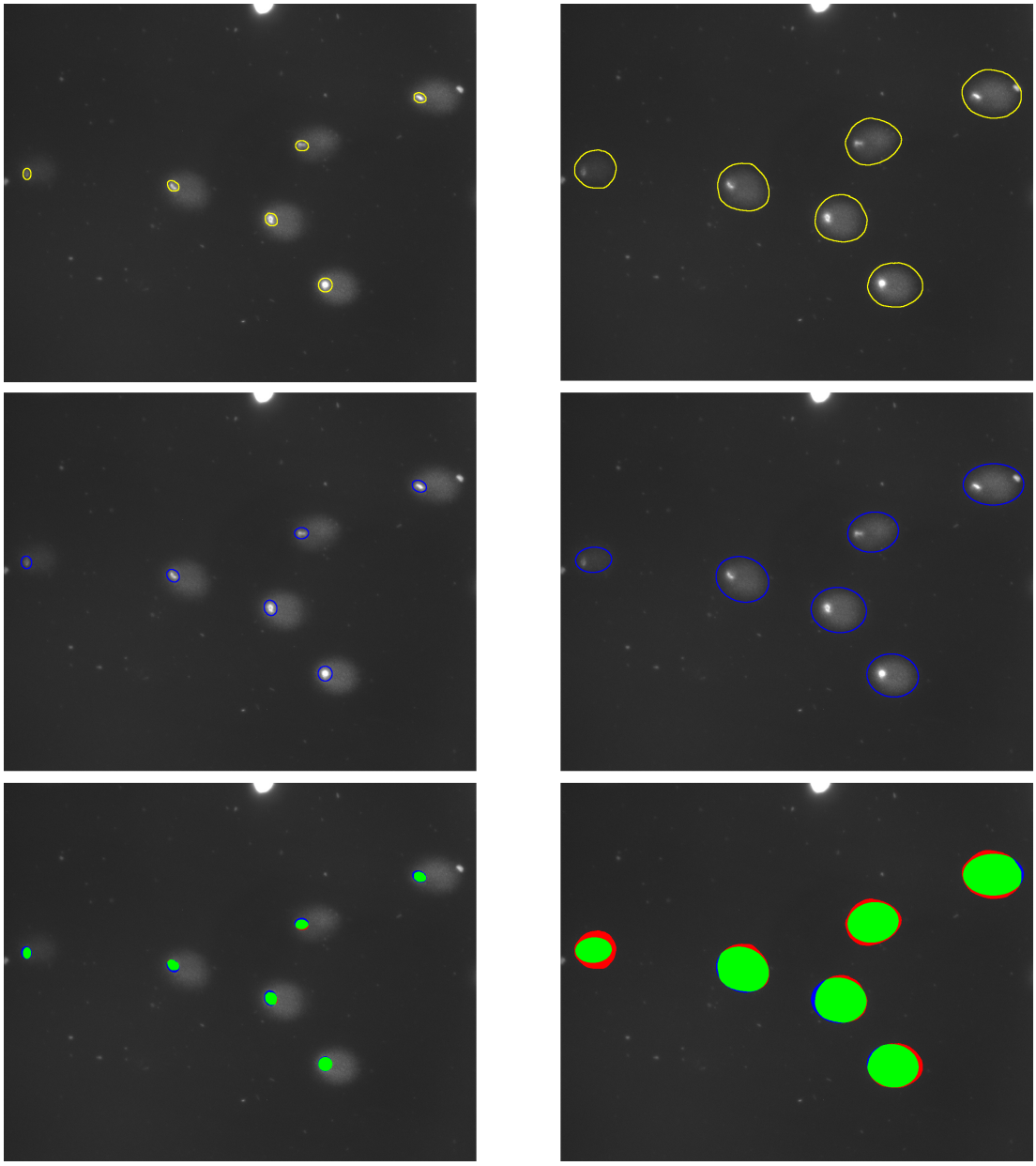


Figura 4.19: En la primera fila y de izquierda a derecha, los contornos segmentados manualmente de las cabezas y colas, respectivamente. En la segunda fila, los contornos segmentados por el algoritmo. En la tercera fila se obtienen los parámetros de la matriz de confusión; los píxeles verdes pertenecen a verdaderos positivos; los azules a falsos positivos; los rojos a falsos negativos; el resto de de la imagen a verdaderos negativos.

Tabla 4.1: Parámetros del algoritmo *FreeComet*.

Parámetro	**Valor**
PREPROCESSING_MEDIAN_RADIUS	6
PREPROCESSING_CLOSING_RADIUS	5
COMET_FILTERING_CLOSING_RADIUS	5
COMET_PROCESSING_DILATION_RADIUS	5
COMET_PROCESSING_MEDIAN_RADIUS	5
COMET_MAXIMUM_HEIGHT_WIDTH_RATIO	1.2
COMET_MAXIMUM_WIDTH_HEIGHT_RATIO	1.4
TAIL_IMPROVAL_ELLIPTICAL_OPENNING_RADIUS	20
HEAD_COMET_AREA_PROPORTION	0.26
COMET_MINIMUM_SIZE_MULTIPLIER	0.001
HEAD_MINIMUM_CONVEXITY	0.85
HEAD_MINIMUM_CIRCULARITY	0.8
HEAD_MINIMUM_SIZE_MULTIPLIER	0.0002

Tabla 4.2: Métricas IoU de las áreas de la cabeza y cola y objetos detectados para los algoritmos OpenComet y FreeComet.

	OpenComet	FreeComet
IoU cola	42.75 %	70.72 %
IoU cabeza	30.06 %	70.14 %
Detección de objetos	76.16 %	97.33 %

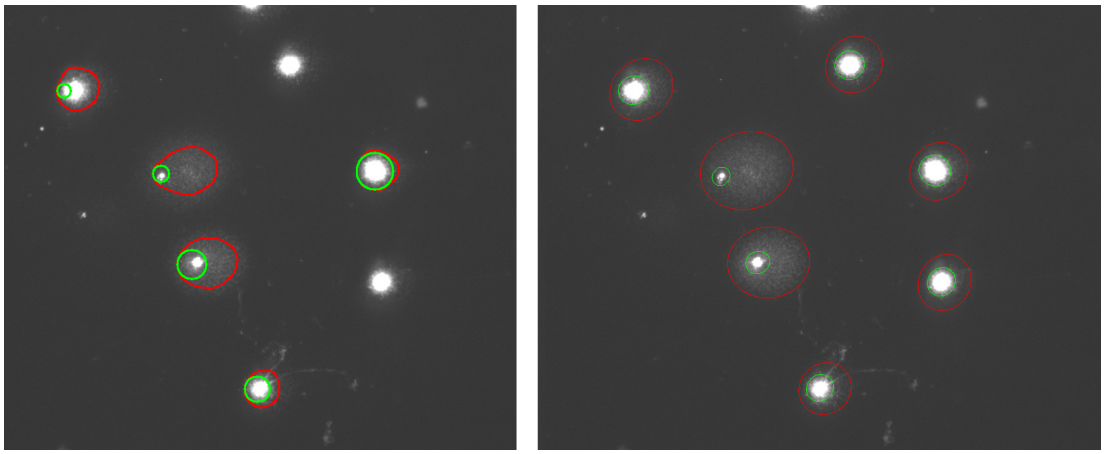


Figura 4.20: Izq.: segmentaciones de cometas de la implementación de OpenComet. Dcha.: segmentaciones de cometas de FreeComet.

Capítulo 5

Aplicación

En este capítulo se expone cómo se ha efectuado el desarrollo de la aplicación del proyecto. Para ello, primero se explica cómo se realizado el análisis. Después se detalla cómo se ha realizado el diseño de la aplicación. Para terminar se profundiza en los detalles de implementación y se indican las pruebas que se han realizado para validar.

5.1 Análisis

Previo al desarrollo de la aplicación *per se* se realizó un análisis en el que se definieron tanto los requisitos funcionales como no funcionales que esta debía satisfacer. Primero se definieron los requisitos funcionales teniendo en consideración los actores que iban a intervenir en el uso de la aplicación. Se determinó que el único actor es el usuario final. En la Figura 5.1 se muestra el diagrama de casos de uso de la aplicación. En el apéndice A se detallan estos casos de uso. El usuario puede:

Gestionar sus proyectos

El usuario puede crear (Tab. A.1), abrir (Tab. A.2) y guardar proyectos (Tab. A.3). En ellos añade las imágenes (Tab. A.4) que tiene interés en segmentar. Las imágenes se pueden eliminar (Tab. A.5) y cambiar su nombre (Tab. A.6). Para trabajar con la máxima visibilidad se le permite trabajar en modo *pantalla completa* (Tab. A.7 y Tab. A.8). También puede rehacer (Tab. A.9) o deshacer la última acción (Tab. A.10) y cambiar el idioma de la aplicación (Tab. A.11). El sistema considera que en los cometas la cabeza está en el lado izquierdo y la cola en el derecho. Por esta razón el usuario tiene la opción de voltear las imágenes horizontalmente en caso conveniente (Tab. A.12). También puede invertir sus colores (Tab. A.13) porque la aplicación asume la parte más oscura como fondo y la más brillante como cometa.

Segmentar sus imágenes

- Automáticamente

  Los usuarios pueden segmentar automáticamente sus imágenes con el algoritmo de segmentación (Tab. A.14). El algoritmo puede ser configurado antes o a la hora de segmentar (Tab. A.15).

- Manualmente

  Con un método de creación de contornos en el visor de imágenes. Los usuarios pueden tanto añadir cometas nuevos (Tab. A.16) como eliminar (Tab. A.17) y editar cometas existentes (Tab. A.18). Los cometas deben tener siempre contorno de la cabeza. El contorno de la cola es opcional. Tanto la creación de los contornos de la cabeza como los de la cola implican añadir (Tab. A.19), conectar (Tab. A.20), mover (Tab. A.21) y eliminar puntos que delimitan la forma de los contornos (Tab. A.22). A la hora de editar un cometa tienen la opción de eliminar directamente la cola si así lo ven oportuno (Tab. A.23). La segmentación manual puede exigir ver las imágenes más de cerca por lo que tienen la opción de acercarlas (Tab. A.24) o alejarlas (Tab. A.25).

Obtener las métricas de los cometas

Para ver las métricas de los cometas el usuario puede generar una hoja de cálculo con todos los cometas segmentados de cada imagen (Tab. A.26) o mirar las métricas específicas de un cometa (Tab. A.27). En ambos casos el sistema calcula las métricas de los cometas requeridos si no lo ha hecho ya.

Con respecto a los requisitos no funcionales no se encontraron dependencias de requisitos de organización o externos como son los éticos o legislativos. Los aspectos más relevantes que la aplicación tuvo en cuenta fueron relacionados con el producto. Estos son principalmente que el sistema fuera fácilmente usable, eficiente y fiable.

5.2 Diseño

La primera tarea en el diseño consiste en definir los elementos que van a componer la interfaz de usuario y su colocación en el espacio. En la Figura 5.2 se muestra la primera maqueta que se creó, la cuál está muy inspirada en la interfaz de OpenComet. La idea inicial siempre fue tener un visor y una lista de imágenes. En esta maqueta, la lista de imágenes viene representada por las pestañas que están en la parte superior del visor, una por imagen cargada en la aplicación. El usuario podría cambiar de imagen seleccionando otra pestaña. Para eliminar una imagen bastaría con eliminar su pestaña. Como toda aplicación de escritorio se incluyó

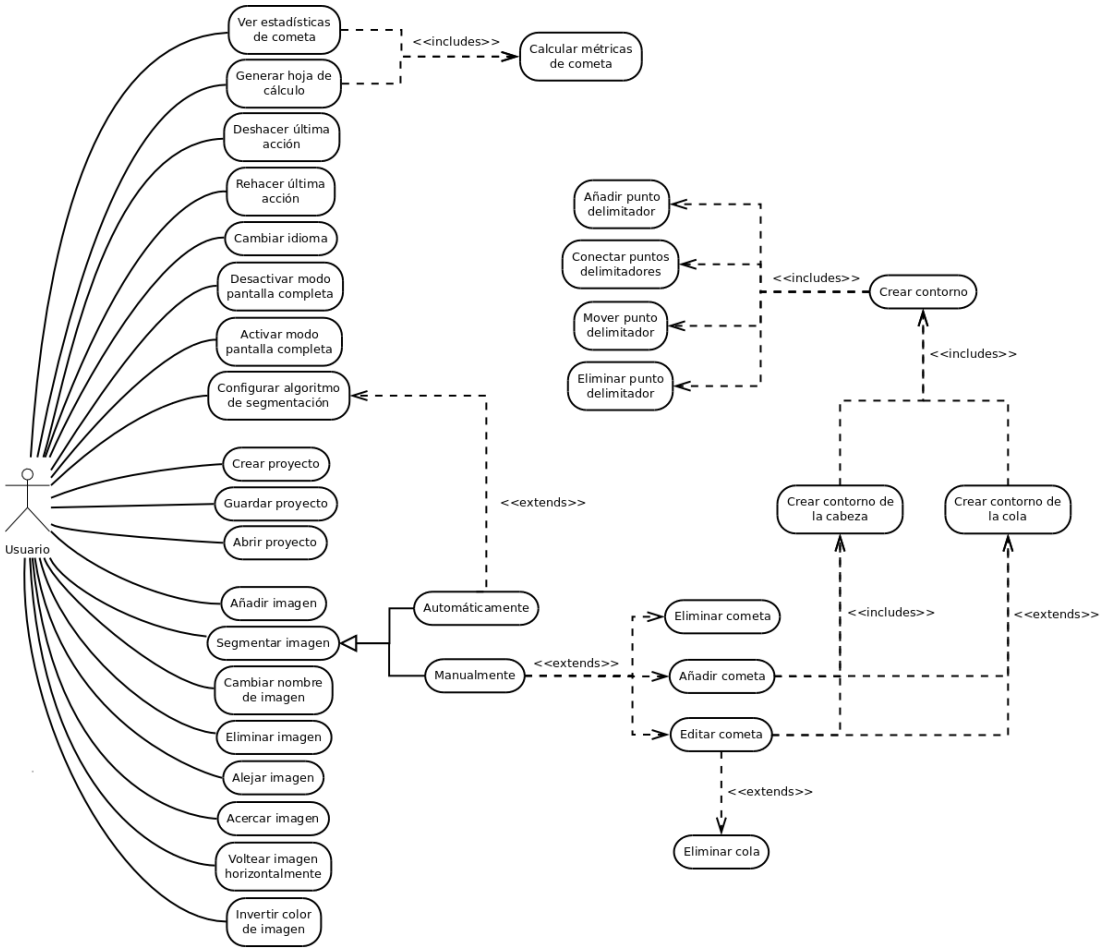


Figura 5.1: Diagrama de casos de uso de la aplicación.

una barra de menú en la que el usuario tendría acceso a funcionalidades genéricas. A la derecha, el usuario dispondría de un área con botones para cargar las imágenes y elegir la ruta para guardar en memoria los resultados del análisis. La configuración del algoritmo de segmentación automático se incluía en esa misma área. Otra idea inicial era tener una ventana de depuración en la parte inferior de la ventana.

No obstante, esta maqueta presentaba dos inconvenientes principales. El primero es que no se estaban incluyendo los componentes encargados de la segmentación manual. El segundo inconveniente es que, a partir de un número no muy elevado de imágenes, la lista de imágenes en formato de pestañas se convertiría en un engorro.

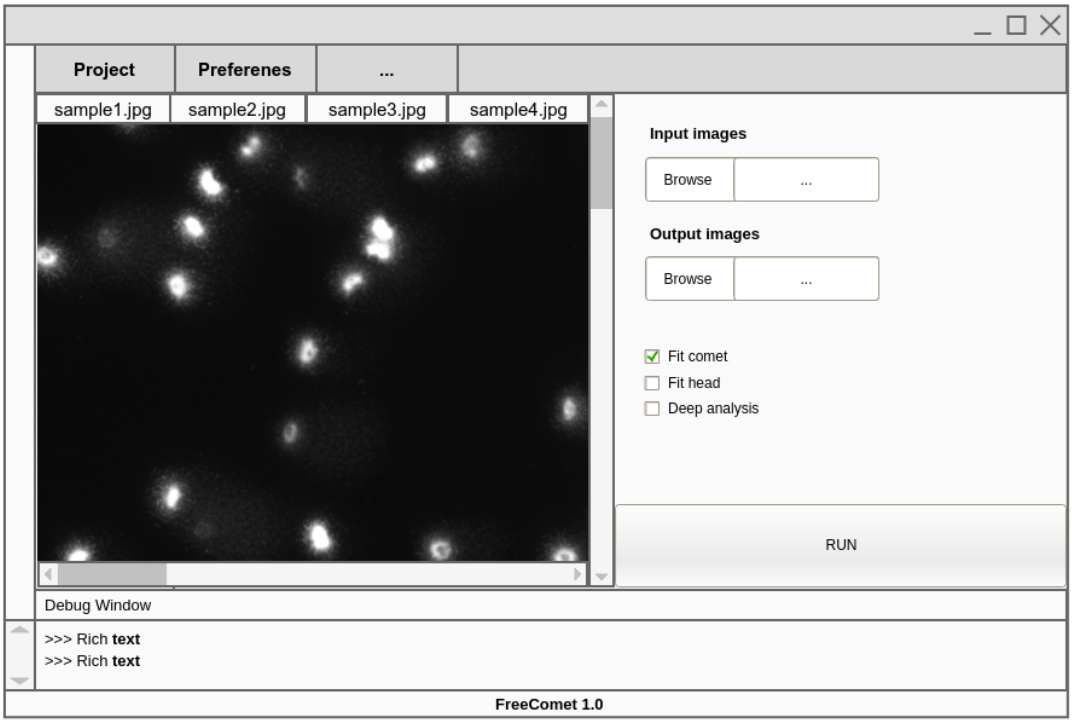


Figura 5.2: Primer *mockup* de la interfaz de usuario.

En la Figura 5.3 se muestra la segunda maqueta que se construyó a raíz de la primera. Las pestañas fueron sustituidas por una lista con barra de desplazamiento vertical a la izquierda de la ventana. Se añadió también una barra de herramientas con las acciones principales que tendría la aplicación. Finalmente se decidió dividir el área del visor en dos partes, incluyendo un panel superior con herramientas que reflejan la posibilidad de trabajar manual y directamente sobre la imagen. Elementos a destacar que se incluyeron son los botones para analizar (segmentar) las imágenes, el de configuración y la herramienta de escalado para el visor de imágenes. El resto de elementos de la primera maqueta no se modificaron.

El siguiente paso fue diseñar la estructura interna de la aplicación. Esta se construyó en

base al patrón arquitectónico Modelo-Vista-Controlador. Esta arquitectura divide la aplicación en tres capas: la capa *modelo*, la capa *vista* y la capa *controlador*. En la capa modelo se implementa la lógica de negocio de la aplicación. En la capa vista se gestiona la renderización de los elementos gráficos. El controlador hace de capa intermedia entre las dos. El principal objetivo de utilizar esta arquitectura es desacoplar la lógica de negocio de su representación gráfica. Existen múltiples formas de implementar esta arquitectura, dándoles mayor peso a unas capas u a otras. El proceso más común se puede ver en la Figura 5.4. Típicamente, el usuario utiliza los elementos gráficos de la capa vista para interaccionar con el sistema y esta notifica al controlador de las acciones que ha realizado. En base a la información proporcionada, el controlador decide qué hacer. Así, puede manipular los datos de la capa modelo o directamente trabajar sobre la capa vista.

Capa modelo

La capa modelo (Fig. 5.5) está constituida por las imágenes que los usuarios cargan en la aplicación. Estas imágenes se almacenan en la clase *Model* como objetos de la clase *Sample*. La clase *Model* permite la adición y eliminación de estos objetos según lo solicite la capa controlador. Los objetos de la clase *Sample* se diferencian por un identificador único y almacenan la imagen y el nombre de este. Como cada imagen tiene que mantener un registro con los cometas que se segmenten en ella, los objetos de *Sample* tienen objetos asociados de la clase *Comet* que representan los cometas. Estos se definen con un identificador único y de ellos se registra el contorno de la cola y el de la cabeza. Cada cometa tiene una instancia de la clase *CometParameters* donde se calculan y almacenan sus métricas cuando así lo requiere el sistema.

La capa modelo utiliza el paquete de terceros *numpy* (NumPy) y el módulo *utils* (Fig. 5.6). *numpy* dispone de múltiples métodos para manipular matrices que es como se representan las imágenes y contornos en código. $utils$ es un módulo auxiliar con diversos métodos, principalmente de manipulación de imágenes y contornos y matemáticos para el cálculo geométrico.

Capa vista

La capa vista (Fig. 5.7) se compone de todos los elementos gráficos de la aplicación. *View*, la clase principal, se encarga de gestionar su funcionamiento interno. Estos componentes deben indicar al usuario de qué herramientas dispone para interaccionar con la aplicación y trabajar con las imágenes del modelo.

La aplicación está compuesta por distintas ventanas. Por cada ventana de funcionalidad relevante en el sistema se creó una clase específica. Estas ventanas extienden de la clase *BaseWindow* para heredar el comportamiento común. La ventana principal de la aplicación es

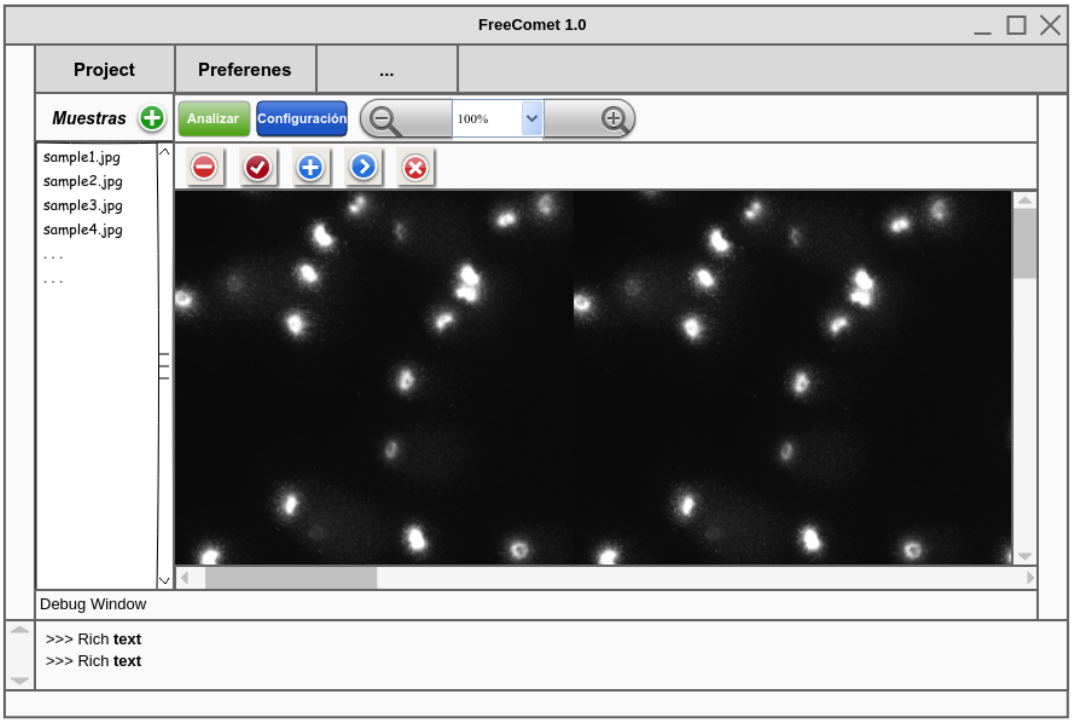


Figura 5.3: Segundo *mockup* de la interfaz de usuario.

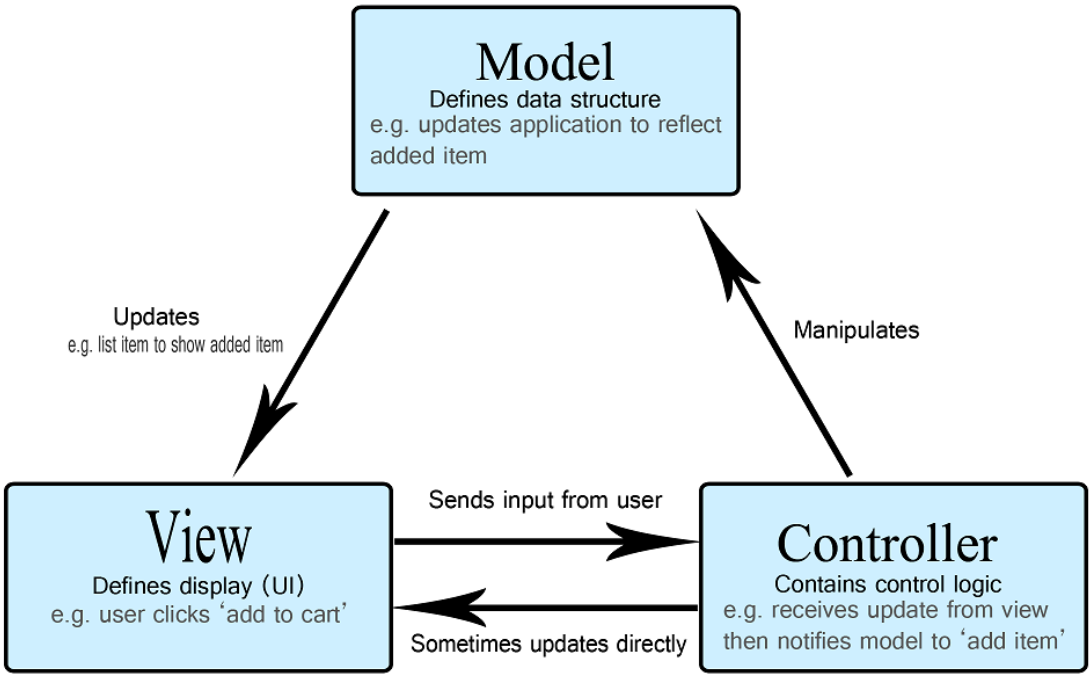


Figura 5.4: Flujo estándar de la arquitectura Modelo-Vista-Controlador.

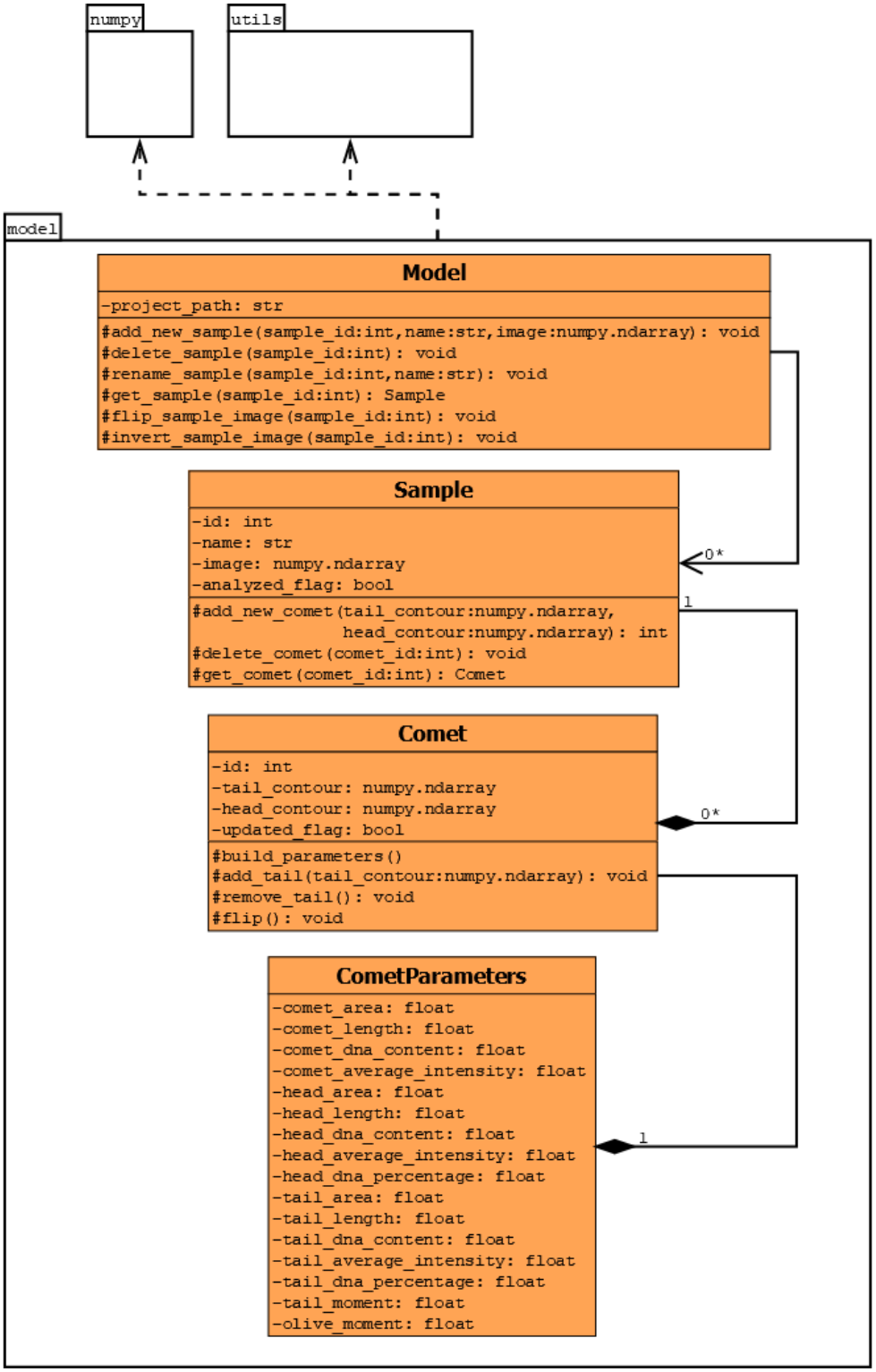


Figura 5.5: Diagrama UML de la capa *modelo*.

Figura 5.6: Diagrama UML del módulo *utils*.

MainWindow. La ventana principal cuenta con una barra de menú, una barra de herramientas, una lista con las imágenes del modelo y un visor con herramientas para trabajar sobre las imágenes. Así, para gestionar estos elementos se crearon las clases *MenuBar*, *ToolBar*, *SamplesView* y el módulo *canvas*, respectivamente. *MainWindow* también contiene una ventana interna que se activa cuando un cometa es seleccionado. Esta ventana se representa en la clase *SelectionWindow*. La ventana para visualizar las métricas concretas de un cometa es *CometParametersWindow*.

La configuración por defecto del algoritmo de segmentación la establecen los usuarios desde la ventana *MainSettingsWindow*. Para segmentar las imágenes se utiliza la ventana *AnalyzeSamplesWindow*, la cual permite una configuración temporal del algoritmo para una segmentación concreta. Por esta razón, ambas ventanas heredan los atributos y comportamiento de la clase *SettingsWindow*. La configuración inicial de estas ventanas se establece en base a los parámetros de la instancia de la clase *controller.AlgorithmSettings*.

AnalyzeSamplesWindow también posee una instancia de *AnalyzeSamplesView*. Tanto *SamplesView* como *AnalyzeSamplesView* extienden de *ListView*, la clase predeterminada destinada a renderizar un modelo genérico con funciones CRUD. La finalidad de *SamplesView* es mostrar las imágenes que hay cargadas en un determinado momento en el modelo mientras que la de *AnalyzeSamplesView* es la de permitir a los usuarios seleccionar las imágenes que quieren segmentar del conjunto total existente.

LoadSamplesWindow y *AnalyzeSamplesLoadingWindow* son ventanas que se utilizan para informar al usuario del progreso en la ejecución de la carga de imágenes en el sistema y de la segmentación automática de las imágenes. La aplicación también cuenta con varios diálogos que se utilizan directamente a través del paquete GTK. Estos diálogos retornan un identificador como respuesta en base a lo que haya seleccionado el usuario. En la clase estática *DialogResponse* se hace un mapeado entre estos identificadores con su significado lógico.

La aplicación únicamente soporta la segmentación manual de una imagen de forma simultánea. Por esta razón, para saber sobre qué imagen se está trabajando, *View* almacena el identificador de la imagen *activa*. Los usuarios pueden modificar los parámetros de los elementos gráficos (p. ej. cambiar el valor del cuadro combinado de ampliación, mover las barras de desplazamiento del visor de imágenes, etc.). Puesto que la configuración de estos parámetros es única para cada imagen, cuando se activa una imagen es necesario recuperar la configuración de los parámetros según estaban previamente establecidos. Por esta razón la capa vista utiliza el módulo *view_store* (Fig. 5.8).

view_store guarda tanto la información necesaria de las imágenes para la capa modelo como la configuración de los parámetros de los componentes de la capa vista en objetos de *SampleParameters*. Estos objetos se almacenan en *ViewStore*, la clase principal del módulo. *ViewStore* extiende de la clase *Observable* del módulo *observer* donde se implementa el patrón

de diseño *observador* [45]. Así, cuando hay cambios en su estado informa a todos los observadores registrados. En la aplicación el único observador es *View*, que implementa la interfaz *observer.Observer* y actúa en base a los cambios que se producen en *ViewStore*.

Cada instancia de *SampleParameters* guarda la información necesaria para trabajar en el visor de imágenes en un objeto de *CanvasParameters* y para la herramienta de ampliación se guarda en un objeto de *ZoomToolParameters*.

En *ZoomToolParameters* se almacena el modelo de la herramienta de ampliación. La aplicación tiene unos niveles de ampliación por defecto pero también permite al usuario introducir valores arbitrarios de escalado. Por esta razón es necesario registrar un modelo de escalado para cada imagen. El otro atributo que se guarda es el índice activo del modelo para saber qué nivel de ampliación está en uso.

En *CanvasParameters* se almacenan atributos como la imagen *per se* que está siendo renderizada en el visor y la posición de las barras de desplazamiento. La información necesaria para dibujar los cometas segmentados se guarda en instancias de la clase *CometView*. También se registran todos los contornos de los cometas que están siendo editados y creados manualmente por el usuario. Estos contornos aún no forman parte del modelo. Para representar estos contornos se creó la clase *CanvasContour*. Estos contornos se construyen a partir de objetos de la clase *DelimiterPoint*. Como hay contornos de cabeza y de cola, los objetos de la clase *DelimiterPoint* tienen un atributo que especifica a qué tipo de contorno pertenecen mediante la utilización de la clase estática *DelimiterPointType*.

La principal finalidad del módulo *canvas* (Fig. 5.9) es mostrar las imágenes en el visor , dibujar los contornos de los cometas y permitir la edición y creación de estos. Su clase principal *Canvas* tiene una instancia de *ZoomTool*, *ColorTool* y *Brush*. *ZoomTool* es la herramienta de ampliación de imágenes. *ColorTool* es la herramienta que permite al usuario escoger el color con el que quiere que los contornos de los cometas se dibujen. *Brush* es el pincel que utiliza *Canvas* para dibujar en el visor. Otra parte muy importante de *Canvas* es su *estado*. Al usuario se le permite trabajar sobre el visor de imágenes a través de una implementación del patrón de diseño *estado* [45]. Todas las clases *estado* extienden de la clase *State*, que a su vez extiende de *Singleton* donde se implementa el patrón *instancia única* [45]. Las instancias que *Canvas* puede tener como estado son *CanvasSelectionState* y *CanvasEditingState*. Ambas clases extienden de la clase abstracta *CanvasState*.

La clase *CanvasSelectionState* facilita el desplazamiento por el visor de imágenes y permite seleccionar los cometas segmentados. *CanvasEditingState* se utiliza explícitamente para la segmentación manual. *CanvasEditingState* también posee estado al que delega ciertas funciones. Las instancias que puede tener como estado son *EditingSelectionState* y *EditingBuildingState*. Ambas heredan de *ActionState*, que a su vez hereda de *CanvasState*.

EditingSelectionState permite la selección de objetos *DelimiterPoint* y puede disparar los

casos de uso *Eliminar punto delimitador*, *Mover puntos delimitadores* y *Añadir punto delimitador*. Para seleccionar los puntos *DelimiterPoint*, puede utilizar la clase *SelectionArea* para crear un área de selección en el visor. Los puntos que se seleccionan se almacenan en *DelimiterPointSelection* como instancias de *SelectedDelimiterPoint*. Para añadir un nuevo *DelimiterPoint* utiliza la clase *RequestedDelimiterPoint*.

EditingBuildingState permite disparar los casos de uso *Añadir punto delimitador* y *Conectar puntos delimitadores*. *EditingBuildingState*, al igual que sus predecesores, tiene un estado que varía en base al tipo de contorno que se esté construyendo. Si el usuario construye el contorno de la cola del cometa, el estado de *EditingBuildingState* es *BuildingTailContourState*. Si por el contrario construye el de la cabeza, el estado es una instancia de *BuildingHeadContourState*.

Del módulo *utils* se emplean métodos de escalado de coordenadas necesarios para dibujar los contornos de los cometas a distintas escalas. Del paquete *GdkPixbuf* se utilizan los objetos *Pixbuf* para renderizar las imágenes en el visor de imágenes. *Gtk* (GTK) y *Gdk* (GDK) son módulos necesarios que van de la mano y que se encargan tanto de la renderización como del comportamiento de todos los elementos gráficos. *Pycairo* lo utiliza *canvas* para para el dibujado en el visor de imágenes.

Capa controlador

La clase principal de la capa controlador (Fig. 5.13) es *Controller*. Esta clase tiene una instancia de la clase *model.Model* y otra de la clase *view.View*, las clases principales del resto de capas, respectivamente. Cuando el usuario dispara alguno de los casos de uso que el sistema soporta desde la capa vista, el método respectivo se invoca en esta clase y se decide cómo proceder. Así, para algunos casos de uso bastará con invocar métodos de la capa vista (p. ej. el caso de uso *Activar pantalla completa*) y para otros será necesario redirigir la petición al modelo (p. ej. el caso de uso *Segmentar imágenes*, que se modela en dos partes en el diagrama de secuencia de la Figura 5.10 y Figura 5.11).

La clase *Controller* tiene una instancia de la clase *AlgorithmSettings* donde se guarda la configuración del algoritmo de segmentación. En ella se almacena el identificador del algoritmo seleccionado y la configuración de sus parámetros. Esta configuración es la que se proporciona por defecto en la segmentación automática de las imágenes.

La segmentación automática de las imágenes se emplea por medio del módulo *algorithms*. La clase *Controller* mantiene una instancia de uno de los algoritmos que extienden de la clase abstracta *Algorithm*, la cual implementa también el patrón *instancia única*. Estos algoritmos son las clases *OpenComet* y *FreeComet*. Así, el usuario puede elegir entre dos algoritmos de segmentación por medio del patrón *estrategia* [45]. *OpenComet* no utiliza parámetros para respetar el procedimiento del algoritmo original. *FreeComet* tiene dos parámetros opcionales y una instancia de *DecisionTree*, el árbol de decisión que implementa la interfaz *Classifier*.

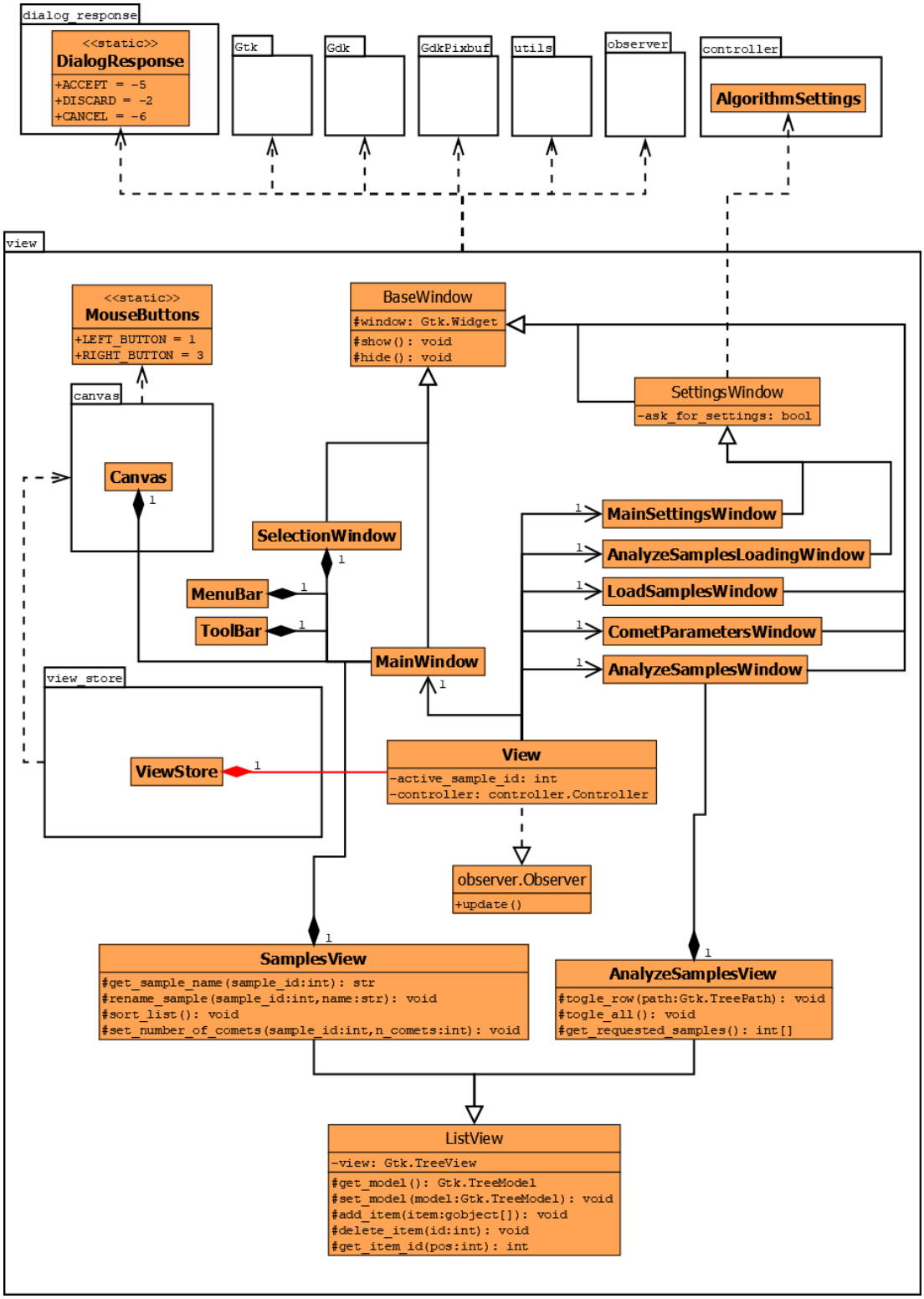


Figura 5.7: Diagrama UML de la capa *vista*.

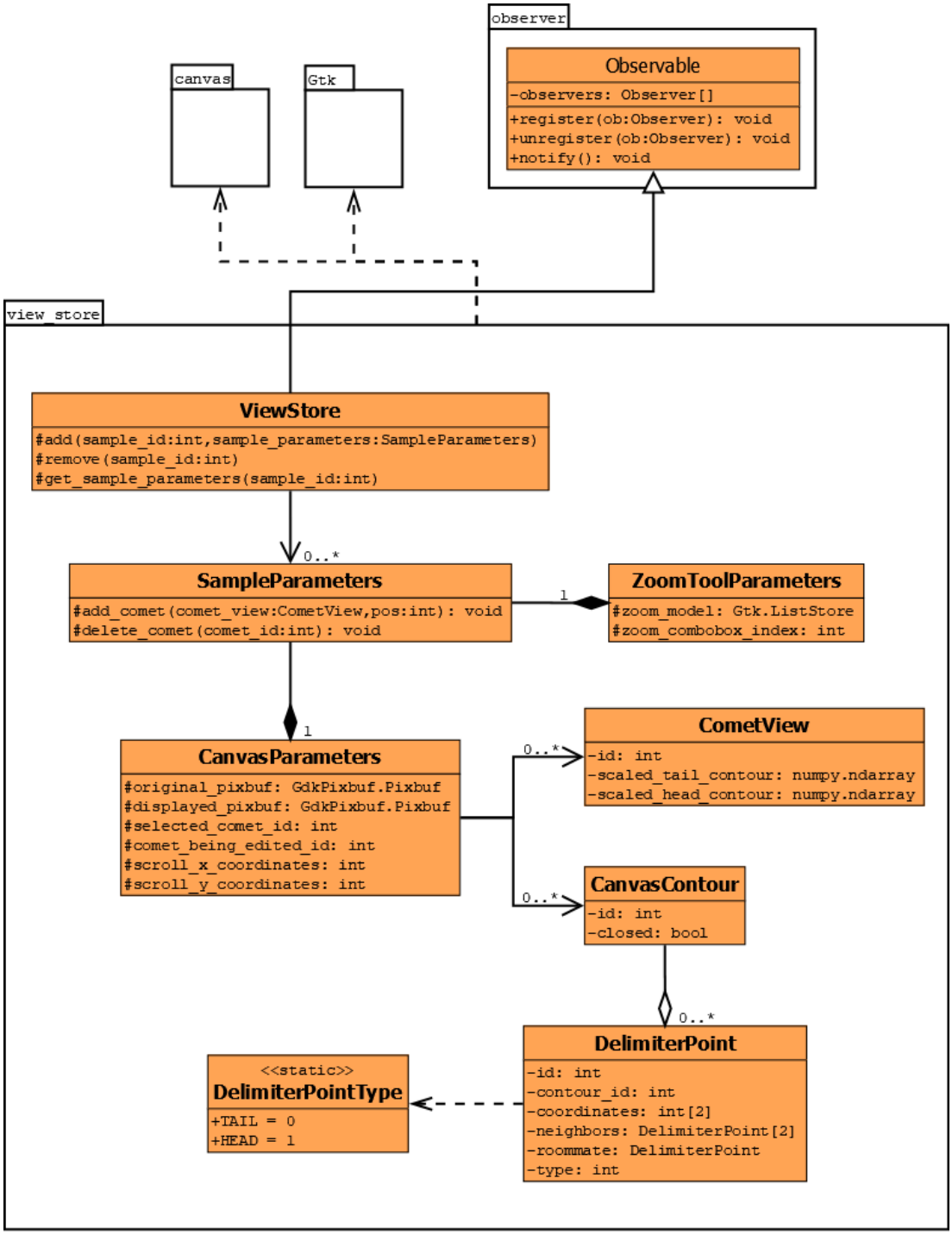


Figura 5.8: Diagrama UML del módulo *view_store*.

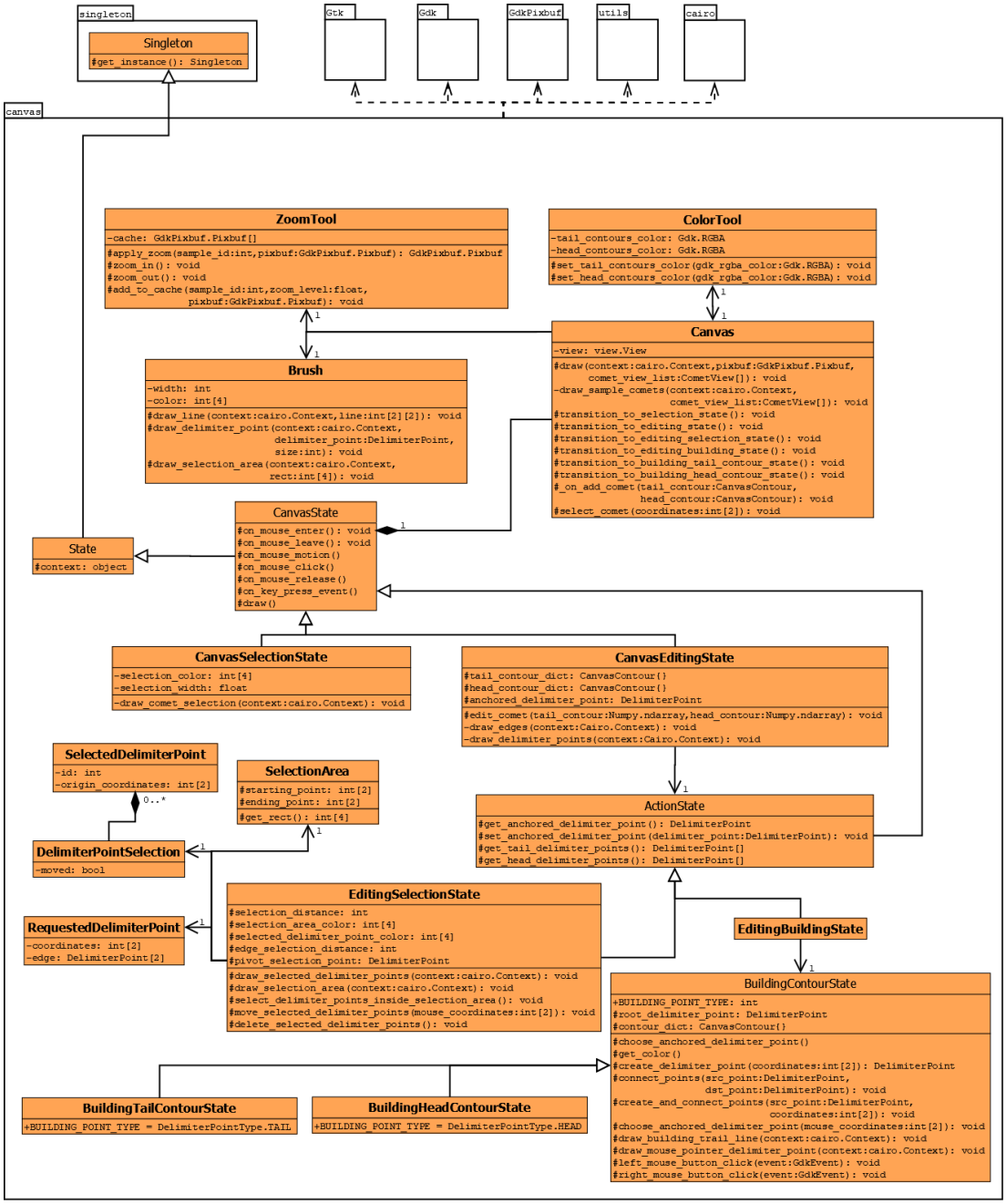


Figura 5.9: Diagrama UML del módulo *canvas*.

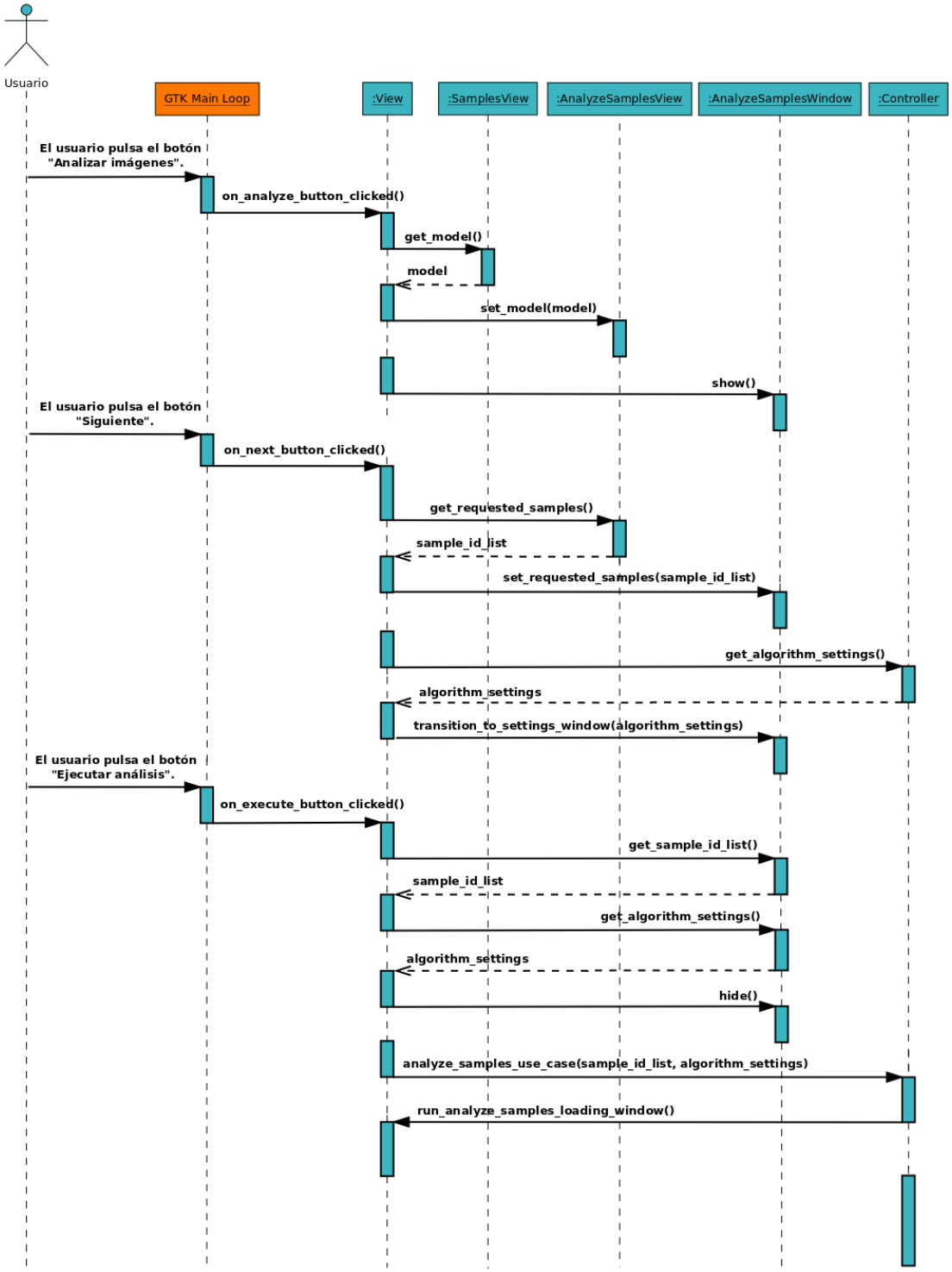


Figura 5.10: Diagrama de secuencia del caso de uso *Segmentar imágenes* (1).

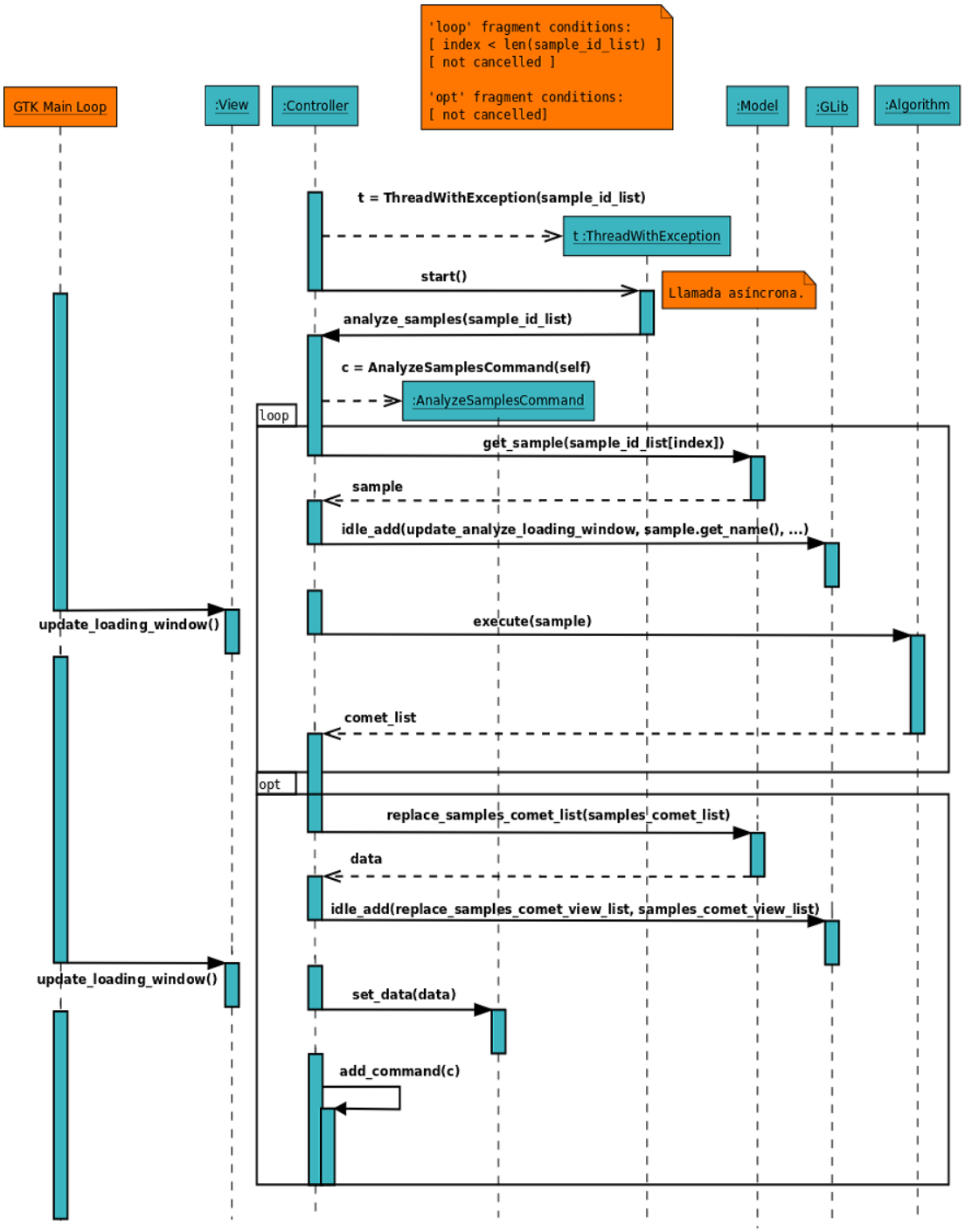


Figura 5.11: Diagrama de secuencia del caso de uso *Segmentar imágenes* (2).

El módulo *algorithms* depende de *image_processing_facade* (Fig. 5.12), un módulo que implementa el patrón de diseño *fachada* [45] a través de su clase estática *ImageProcessingFacade*. Esta clase ofrece distintos métodos de procesamiento de visión artificial mediante el empleo conjunto de los paquetes *utils*, *numpy*, *cv2* (OpenCV) y *skimage* (scikit-image). También utiliza *constants*, un pequeño recipiente de constantes para mapear valores con su interpretación semántica.

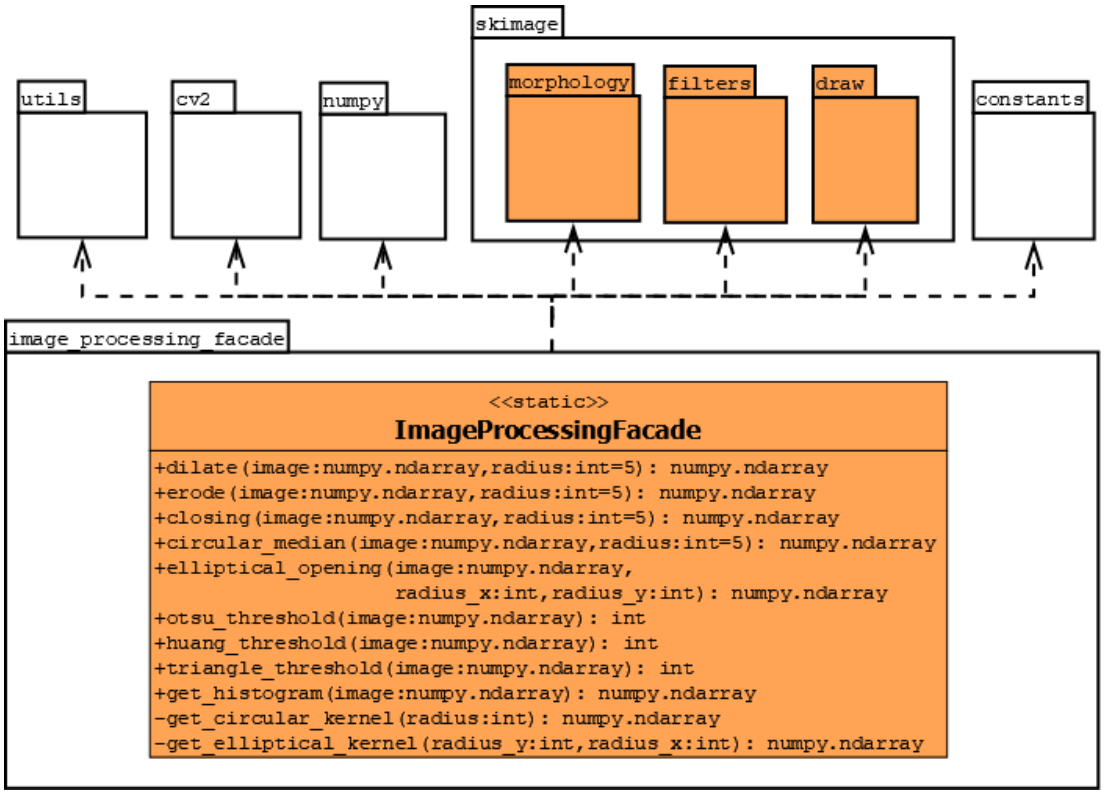


Figura 5.12: Diagrama UML del módulo *image_processing_facade*.

Para guardar y recuperar los proyectos de memoria, el controlador utiliza la clase estática *Parser*. *Parser* también se encarga de generar la hoja de cálculo con las métricas de los cometas.

La internacionalización se consigue por medio de la clase *Strings* que utiliza el módulo de terceros *gettext*. *Strings* almacena todas las cadenas de texto traducibles que usa la aplicación. Cuando el usuario selecciona un idioma, el controlador se encarga de invocar el método de traducción correspondiente, el cuál traduce las cadenas de texto a dicho lenguaje.

La clase *Controller* también tiene dos pilas donde almacena instancias de clases que extienden de la clase abstracta *Command*. Por medio del patrón de diseño *comando* [45] y estas dos pilas se permite al usuario deshacer y rehacer acciones. Cuando se dispara uno de los casos de uso que soporta esta funcionalidad, su *comando* se añade a la pila de *deshacer* (p. ej. *RemoveCometTailCommand* se añade a la pila de *deshacer* cuando se dispara el caso de uso

Eliminar cola de cometa). Si se dispara el caso de uso *Deshacer última acción*, se quita el comando de arriba de la pila y se ejecuta su método *undo()*. A continuación, el mismo objeto se añade a la pila de *rehacer*. Si se disparara el caso de uso *Rehacer última acción* se quitaría el mismo comando de la pila de *rehacer*, se ejecutaría su método *redo()* y finalmente se volvería a añadir a la pila de *deshacer*, y así sucesivamente.

Finalmente, la capa controlador utiliza la clase *ThreadWithException*, una clase que extiende de la clase *threading.Thread* para actualizar las ventanas de progreso por medio de un hilo de ejecución. Cuando un objeto de *ThreadWithExceptions* recibe una llamada a su método *raise_exception()*, su hilo asociado finaliza. Así, el usuario tiene la opción de cancelar la ejecución de los hilos. Los objetos de *ThreadWithExceptions* utilizan el paquete *GLib* para pedirle al hilo principal de ejecución que ejecute métodos con líneas de código de GTK. Esto es así porque sólo el hilo que implementa el bucle principal de GTK puede hacer llamadas a la librería. El controlador también utiliza el módulo *dialog_response* para tener acceso al mapeado de la clase estática *DialogResponse*.

5.3 Implementación

A continuación se hará un recorrido por la interfaz gráfica de usuario comentando los detalles de implementación del sistema. La ventana principal de este se puede ver en la Figura 5.14. En la parte superior, junto al nombre de la aplicación (*FreeComet*), se muestra el nombre del proyecto. El asterisco informa al usuario que ha habido cambios en el proyecto que todavía no se guardaron. Acciones destructivas de estos cambios (crear un proyecto nuevo, abrir otro proyecto o cerrar la aplicación) abrirán un diálogo de confirmación que permite o bien descartarlos o guardarlos.

Desde la barra de menú los usuarios pueden crear un proyecto nuevo, abrir un proyecto existente, guardar el proyecto actual, cambiar el idioma de la aplicación y salir de la aplicación. Debajo está la primera barra de herramientas con funcionalidades comunes a una aplicación de escritorio. Desde ella y de izquierda a derecha, al igual que en la barra de menú, el usuario puede crear un proyecto nuevo, abrir un proyecto existente o guardar el proyecto actual. También puede deshacer/rehacer la última acción y activar/desactivar el modo *pantalla completa*.

Debajo y a la izquierda está la lista de imágenes. Para cada imagen se almacena el nombre y el número de cometas segmentados en ella. Esta etiqueta con el número de cometas segmentados sólo se imprime si la imagen ha sido previamente analizada automática o manualmente. Así, imágenes sin esta etiqueta es una indicación de que no han sido analizadas. Las imágenes se pueden ordenar alfabéticamente haciendo clic en la etiqueta *Imágenes*. Para añadir imágenes a la lista el usuario puede pulsar el botón a la derecha de la etiqueta o haciendo clic con el botón derecho del ratón en un espacio vacío en la lista. Mediante la segunda opción se abrirá

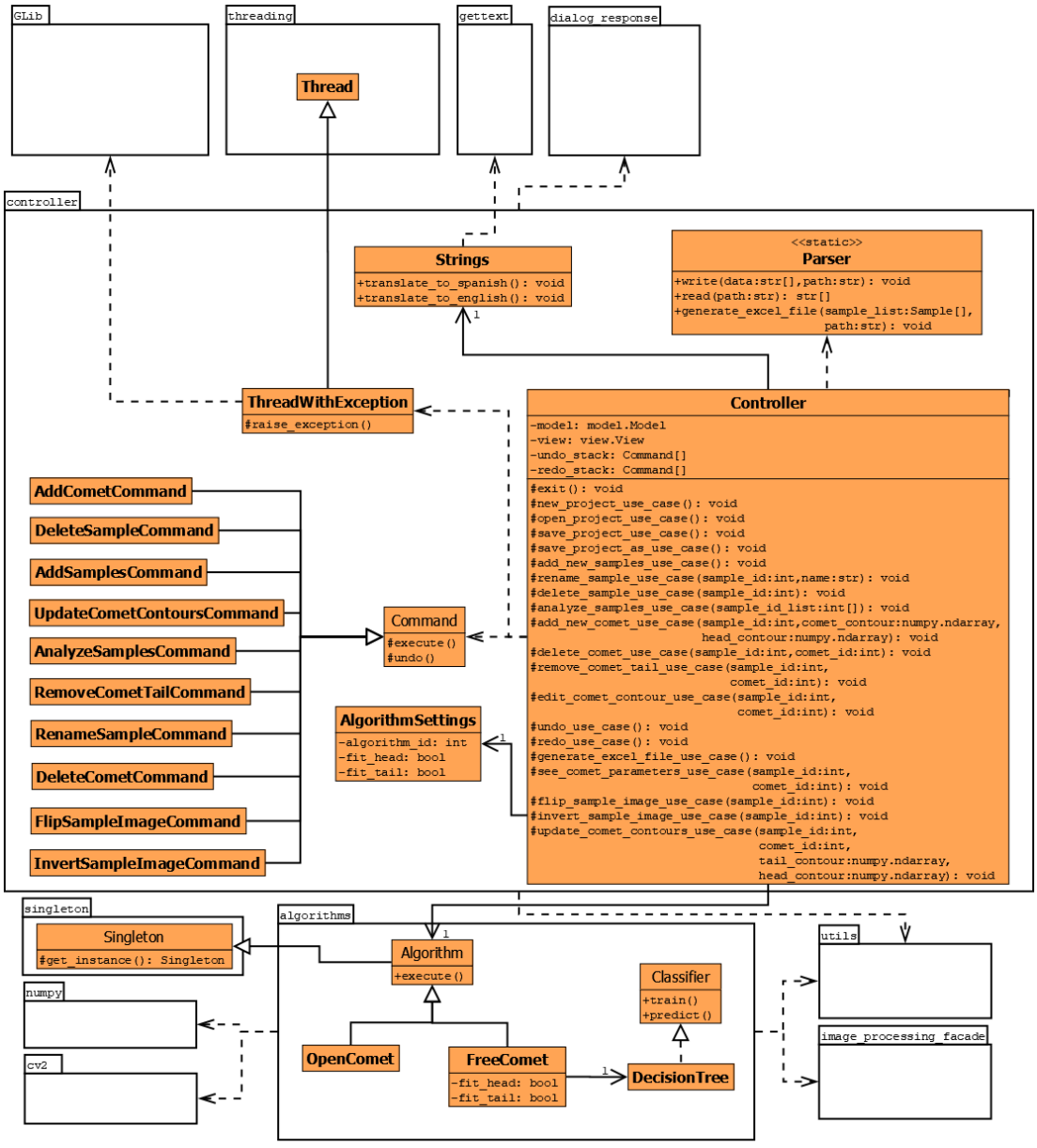


Figura 5.13: Diagrama UML de la capa *controlador* y el módulo *Algorithms*.

un menú contextual con la acción *Añadir* que el usuario podrá seleccionar. Para ambos casos se abre un diálogo donde se eligen las imágenes que se desean cargar. Mientras las imágenes se cargan en la aplicación se muestra una ventana como la de la imagen de la izquierda de la Figura 5.15. El número de imágenes cargadas en la aplicación se imprime debajo de la lista en una etiqueta informativa.

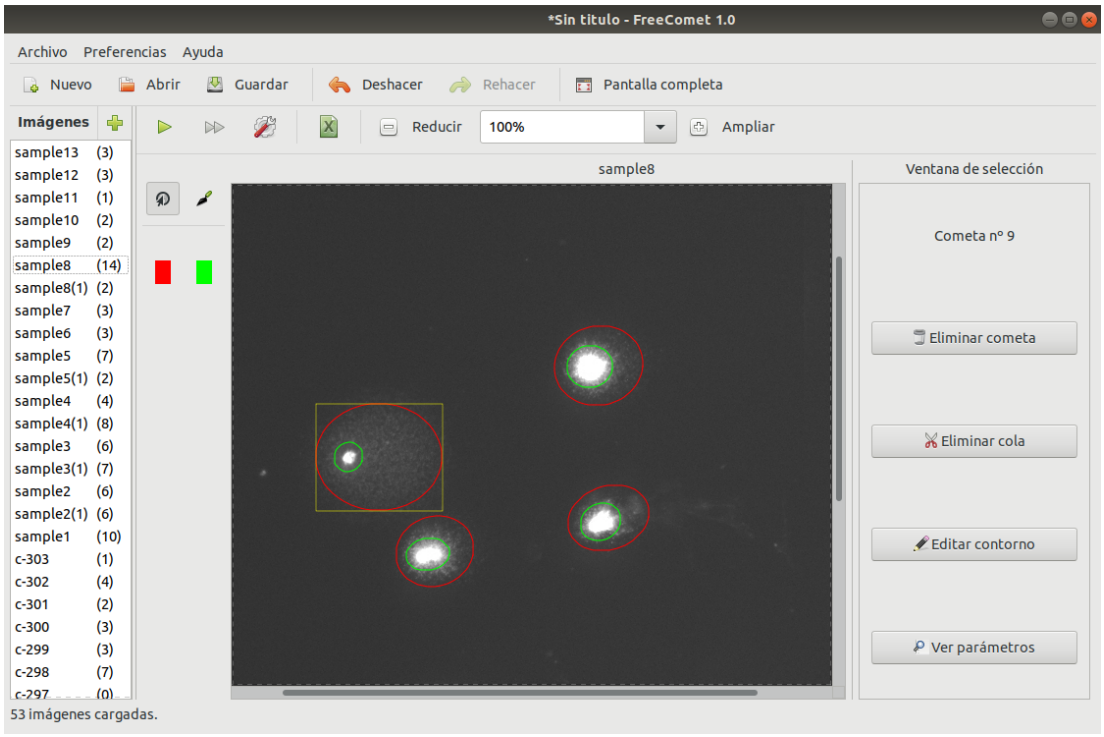
A la derecha de la lista de imágenes y debajo de la primera barra de herramientas se encuentra una segunda barra de herramientas con funcionalidades más específicas a la lógica de la aplicación. Debajo de esta está localizado el visor de imágenes junto a las herramientas de segmentación manual donde los usuarios pueden ver y manipular la imagen seleccionada. A la derecha del visor se encuentra la ventana *Ventana de selección.*

5.3.1 Segmentación automática

El primer grupo de botones de la segunda barra de herramientas constituye la funcionalidad de segmentación automática de las imágenes. El primer y segundo botón sirven para segmentar mientras que el tercer botón para configurar el algoritmo. Si se pulsa en este último se abrirá una ventana de configuración similar a las de la Figura 5.16. En la imagen de la izquierda de la figura se ve la ventana cuando el algoritmo *FreeComet* está seleccionado. El usuario puede escoger que los contornos de la cabeza y cola tengan forma elíptica. Si el usuario escoge *OpenComet* como algoritmo de segmentación la ventana se verá como en la imagen de la derecha. La configuración que se guarde desde esta ventana será la utilizada como predeterminada por el sistema. Por defecto, el algoritmo de segmentación seleccionado es *FreeComet* y sus dos parámetros están desactivados.

El primer botón abre una ventana como la que se muestra en la Figura 5.17. El usuario escoge las imágenes que quiere segmentar marcando sus casillas de verificación. Por defecto, todas las casillas están marcadas. Después de elegir las imágenes se abrirá una ventana de configuración similar a las de la Figura 5.16 con la configuración predeterminada establecida. El usuario puede escoger la configuración del algoritmo que se utilizará en la segmentación pero que no sobrescribirá a la predeterminada. Finalmente se procede con el análisis de las imágenes. Durante el análisis se muestra una ventana de progreso. Esta ventana se puede ver en la imagen de la derecha de la Figura 5.15. Como se puede observar, el usuario puede cancelar el análisis.

La segmentación con el segundo botón de la barra de herramientas consiste en que el sistema selecciona directamente todas las imágenes y las segmenta con la configuración predeterminada del algoritmo.



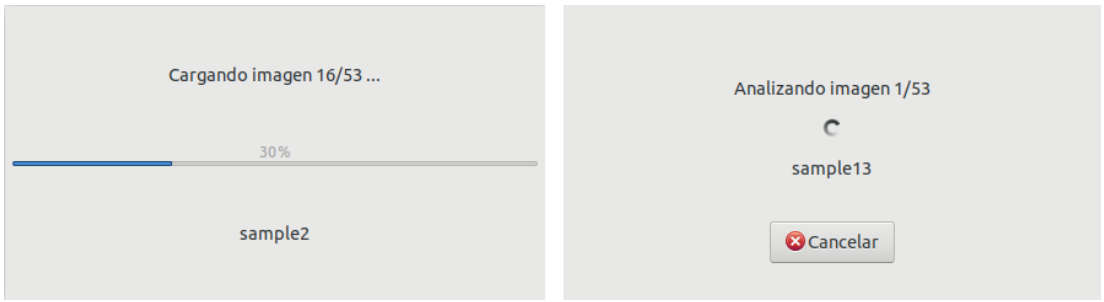


Figura 5.15: Izq.: ventana de progreso al añadir imágenes. Dcha.: ventana de progreso de la segmentación de imágenes.

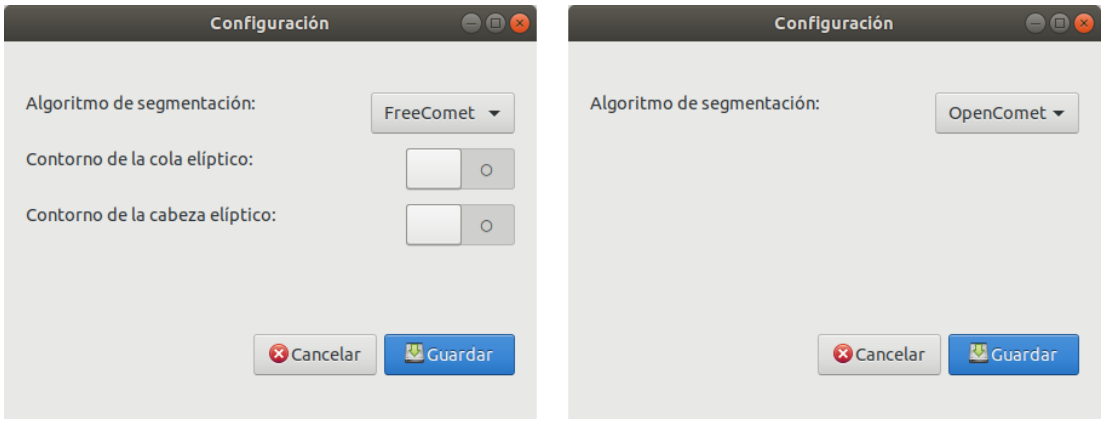


Figura 5.16: Izq.: ventana de configuración del algoritmo de segmentación con el algoritmo *FreeComet* seleccionado. Dcha.: ventana de configuración del algoritmo de segmentación con el algoritmo *OpenComet* seleccionado.

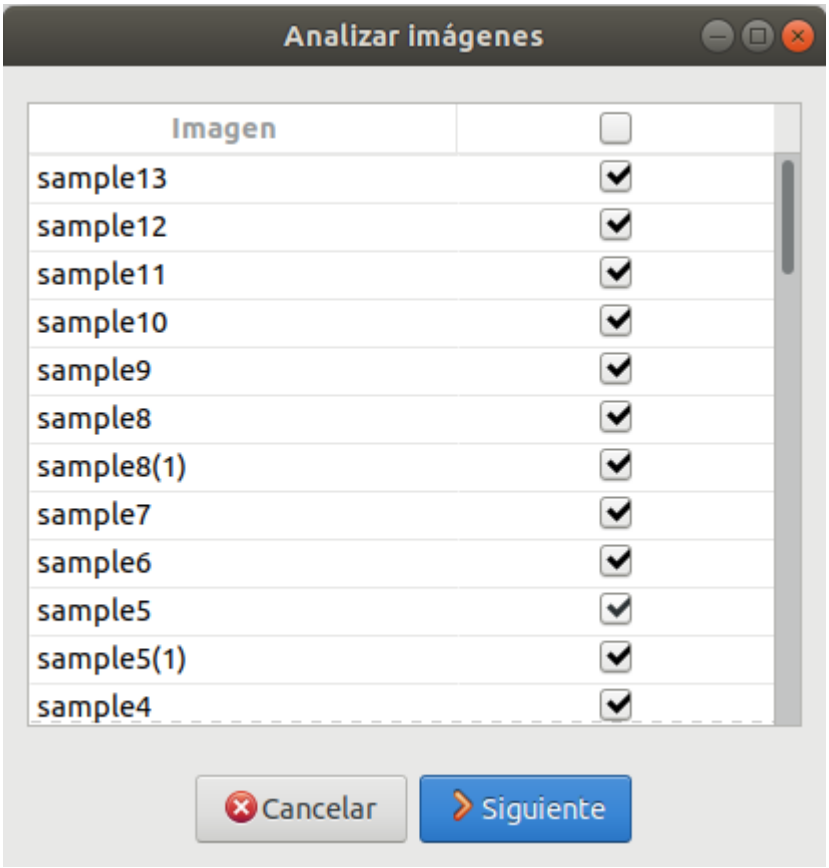
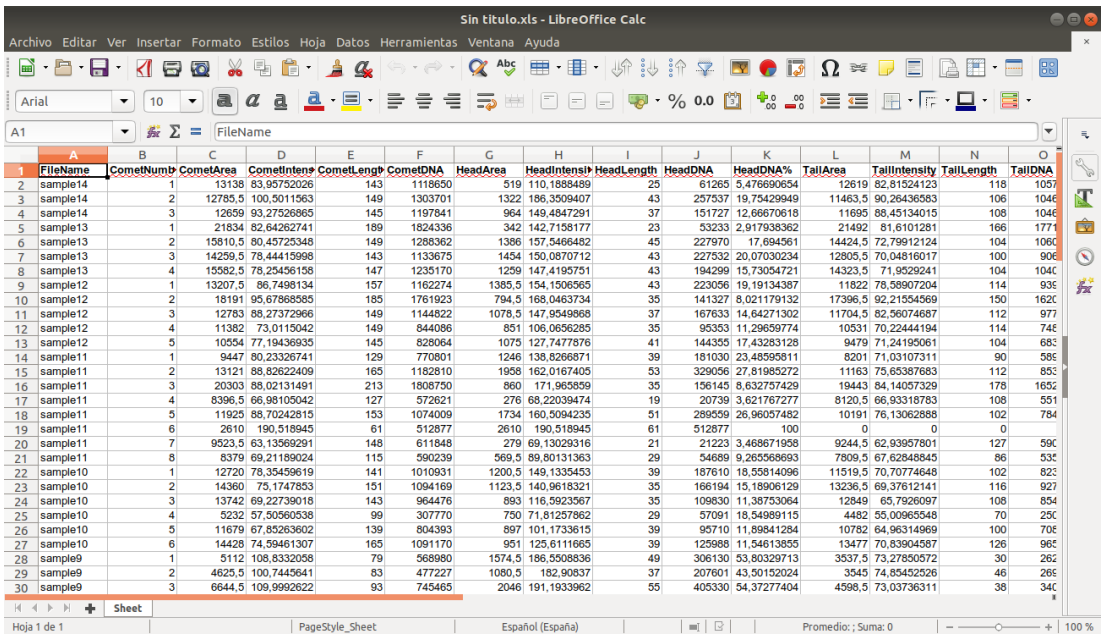


Figura 5.17: Ventana de selección de imágenes para su segmentación.

5.3.2 Métricas de los cometas

A la derecha del botón que configura el algoritmo de segmentación se encuentra el botón que permite generar una hoja de cálculo con las métricas de los cometas de las imágenes segmentadas. En la Figura 5.18 se muestra una imagen de este archivo abierto. Cada fila en la hoja de cálculo es una tupla que contiene el nombre de la imagen al que pertenece el cometa, el número del cometa, el área del cometa, el valor de intensidad promedio en el cometa, la longitud del cometa, la cantidad de ADN en el cometa (suma de las intensidades de los píxeles en el cometa), el área de la cabeza, el valor de intensidad promedio en la cabeza, la longitud de la cabeza, la cantidad de ADN en la cabeza (suma de las intensidades de los píxeles que pertenecen a la cabeza), el porcentaje que ocupa el área de la cabeza, el área de la cola, el valor de intensidad promedio en la cola, la cantidad de ADN en la cola (suma de las intensidades de los píxeles que pertenecen a la cola), el porcentaje que ocupa el área de la cola, el momento de torsión de la cola [46] y el momento de torsión de *Olive* [47].



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	FileName	CometNumb	CometArea	CometIntens	CometLengt	CometDNA	HeadArea	HeadIntensi	HeadLength	HeadDNA	HeadDNA%	TailArea	TailIntensity	TailLength	TailDNA
2	sample14	1	13138	83,95752026	143	1118650	519	110,1888489	25	61265	5,476690654	12619	82,81524123	118	1057
3	sample14	2	12785,5	100,5011563	149	1303701	1322	186,3509407	43	257537	19,75429949	11463,5	90,26436583	106	1046
4	sample14	3	12659	93,27526865	145	1197841	964	149,4847291	37	151727	12,66670618	11695	88,45134015	108	1046
5	sample13	1	21834	82,64262741	189	1824336	342	142,7158177	23	53233	2,917938362	21492	81,6101281	166	1771
6	sample13	2	15810,5	80,45725348	149	1288362	1386	157,5466482	45	227970	17,694561	14424,5	72,79912124	104	1060
7	sample13	3	14259,5	78,44415998	143	1133675	1454	150,0870712	43	227532	20,07030234	12805,5	70,04816017	100	906
8	sample13	4	15582,5	78,25456158	147	1235170	1259	147,4195751	43	194299	15,73054721	14323,5	71,9529241	104	1040
9	sample12	1	13207,5	86,7498134	157	1162274	1385,5	154,1506565	43	223056	19,19134387	11822	78,58907204	114	939
10	sample12	2	18191	95,67868585	185	1761923	794,5	168,0463734	35	141327	8,021179132	17396,5	92,21554569	150	1620
11	sample12	3	12783	88,27372966	149	1144822	1078,5	147,9549868	37	167633	14,64271302	11704,5	82,56074687	112	977
12	sample12	4	11382	73,0115042	149	844086	851	106,0656285	35	95353	11,29659774	10531	70,22444194	114	748
13	sample12	5	10554	77,19436935	145	828064	1075	127,7477876	41	144355	17,43283128	9479	71,24195061	104	683
14	sample11	1	9447	80,23326741	129	770801	1246	138,8266871	39	181030	23,48595811	8201	71,03107311	90	589
15	sample11	2	13121	88,82622409	165	1182810	1958	162,0167405	53	329056	27,81985272	11163	75,65387683	112	853
16	sample11	3	20303	88,02131491	213	1808750	860	171,965859	35	156145	8,632757429	19443	84,14057329	178	1652
17	sample11	4	8396,5	66,98105042	127	572621	276	68,22039474	19	20739	3,621767277	8120,5	66,93318783	108	551
18	sample11	5	11925	88,70242815	153	1074009	1734	160,5094235	51	289559	26,96057482	10191	76,13062888	102	784
19	sample11	6	2610	190,518945	61	512877	2610	190,518945	61	512877	100	0	0	0	
20	sample11	7	9523,5	63,13569291	148	611848	279	69,13029316	21	21223	3,468671958	9244,5	62,93957801	127	590
21	sample11	8	8379	69,21189024	115	590239	569,5	89,80131363	29	54689	9,265568693	7809,5	67,62848845	86	535
22	sample10	1	12720	78,35459619	141	1010931	1200,5	149,1335453	39	187610	18,55814096	11519,5	70,70774648	102	823
23	sample10	2	14360	75,1747853	151	1094169	1123,5	140,9618321	35	166194	15,18906129	13236,5	69,37612141	116	927
24	sample10	3	13742	69,22739018	143	964476	893	116,5923567	35	109830	11,38753064	12849	65,7926097	108	854
25	sample10	4	5232	57,50560538	99	307770	750	71,81257862	29	57091	18,54989115	4482	55,00965548	70	250
26	sample10	5	11679	67,85263602	139	804393	897	101,1733615	39	95710	11,89841284	10782	64,96314969	100	708
27	sample10	6	14428	74,59461307	165	1091170	951	125,6111665	39	125988	11,54613855	13477	70,83904587	126	965
28	sample9	1	5112	108,8332058	79	568980	1574,5	186,5508836	49	306130	53,80329713	3537,5	73,27850572	30	262
29	sample9	2	4625,5	100,7445641	83	477227	1080,5	182,90837	37	207601	43,50152024	3545	74,85452526	46	269
30	sample9	3	6644,5	109,9992622	93	745465	2046	191,1933962	55	405330	54,37277404	4598,5	73,03736311	38	340

Figura 5.18: Hoja de cálculo con las métricas de los cometas segmentados.

El usuario también puede ver las métricas de un cometa concreto seleccionándolo y pulsando en el botón *Ver parámetros* en *Ventana de selección*. Así, se abrirá una ventana con las mismas métricas para el cometa cometa seleccionado (Fig. 5.19).

5.3.3 Segmentación manual

La segmentación manual de la aplicación se efectúa con las herramientas de segmentación manual que se sitúan entre el visor de imágenes y la lista de imágenes. En la imagen de la

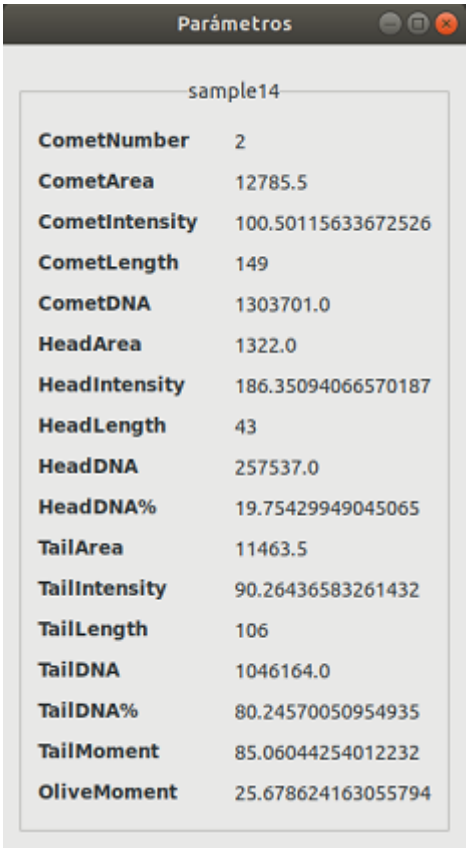


Figura 5.19: Ventana que muestra las métricas de un cometa concreto.

Figura 5.20 se muestran estas herramientas desplegadas. Los botones *Color de la cola* y *Color de la cabeza* son para seleccionar el color con el que se quiere que el sistema dibuje los contornos de la cola y la cabeza de los cometas, respectivamente. El resto de componentes activa un estado distinto y jerárquico en el sistema.

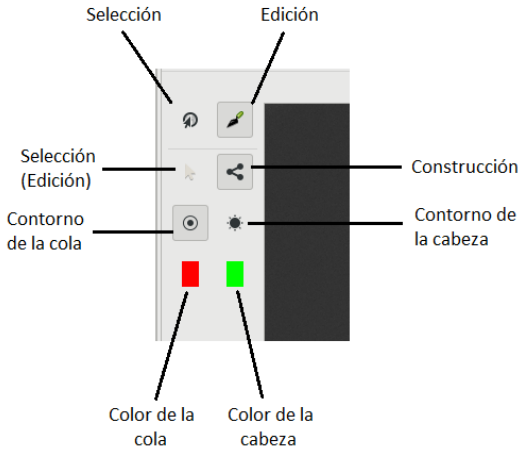


Figura 5.20: Herramientas de segmentación manual de las imágenes.

El estado por defecto de la aplicación es *Selección*. Desde este estado el usuario puede seleccionar los cometas segmentados. Como el lector puede apreciar en la imagen de la Figura 5.14, uno de los cometas está seleccionado. El sistema dibuja un recuadro amarillo alrededor del cometa que ha sido seleccionado. Desde este estado, los usuarios también pueden desplazarse por el visor de imágenes manteniendo pulsado el botón izquierdo del ratón. Además, el resto de elementos se mantienen escondidos menos *Edición*.

Edición es el estado por el que se puede sustituir a *Selección*. Desde este estado los usuarios pueden editar y crear sus propios cometas. Al activarse, el resto de componentes se vuelven visibles. Dentro de *Edición* el usuario puede seleccionar *Selección (Edición)* o *Construcción*.

Construcción es la herramienta que permite a los usuarios segmentar manualmente desde cero los cometas. Una vez en *Construcción*, el usuario activa o bien el botón *Contorno de la cola* o el botón *Contorno de la cabeza* para crear puntos delimitadores del contorno de la cola o la cabeza, respectivamente. El estado por defecto es *Contorno de la cola*.

Para crear puntos delimitadores el usuario debe situar el puntero del ratón en el visor de imágenes. Debajo del puntero se dibuja un punto delimitador indicativo para que el usuario vea dónde se crearía un nuevo punto. El color de este punto depende del tipo de contorno a construir que se haya seleccionado. Para crear un punto, el usuario debe pulsar el botón izquierdo del ratón dentro del visor. Tras la creación del nuevo punto, este pasa a ser el punto

origen. Cuando un punto es punto origen, la acción que se origina al presionar el botón izquierdo del ratón adopta un comportamiento diferente. Así, si el usuario vuelve a hacer clic en otra posición del visor con un punto origen activo se creará un nuevo punto pero además se conectará con el punto origen. Para saber si un punto origen está activo basta con fijarse en si se dibuja una línea temporal entre el punto delimitador indicativo del puntero del ratón y alguno del resto de puntos en el visor. Para que un punto deje de ser punto origen, el usuario puede hacer clic derecho en el visor de imágenes.

El usuario puede elegir que uno de los puntos sea punto origen sin crear un punto nuevo. Para ello puede aproximar el puntero del ratón a uno de los puntos existentes hasta que se cree un *punto de anclaje*. El usuario sabrá si se habrá creado un punto de anclaje porque el color del punto al que se ancla el puntero del ratón pasa a ser amarillo. Además, el punto delimitador indicativo del puntero del ratón desaparece. Si el usuario pulsa el botón izquierdo del ratón cuando un punto delimitador está en anclaje, este pasará a ser el punto origen. Si un punto ya es origen y el usuario efectúa la misma acción, los dos puntos se conectarán entre si.

La conexión de puntos se realiza sólo si el punto origen y el punto destino son del mismo tipo de contorno. En caso de pertenecer a tipos distintos, lo que ocurrirá es que se creará un punto en las mismas coordenadas que el punto destino pero del tipo del punto origen. Si un punto de contorno de cola y otro punto de contorno de cabeza comparten las mismas coordenadas se dibuja un único punto pero circular y con el color del contorno de la cabeza.

Desde *Selección (Edición)* los usuarios pueden seleccionar puntos delimitadores de cualquier tipo. Por esta razón, en este estado los elementos *Contorno de la cola* y *Contorno de la cabeza* se desactivan pero se mantienen visibles. Los puntos delimitadores seleccionados se dibujan de color amarillo. Para seleccionar puntos lo pueden hacer de forma individual pulsando el botón izquierdo del ratón cuando el puntero esté encima de uno. Si mantienen la tecla *Ctrl* presionada mientras hacen clic, podrán anidar el punto a la selección como si de selección de ficheros se tratara. Otra forma de seleccionar múltiples puntos es creando un recuadro de selección manteniendo el botón izquierdo del ratón. Este recuadro se dibuja de color amarillo. Los puntos en el interior del recuadro se dibujan de color amarillo para que el usuario sepa qué puntos se seleccionarán si deja de mantener el botón del ratón. Los puntos seleccionados pueden ser eliminados. Esto se consigue presionando la tecla *Supr* o pulsando el botón derecho del ratón encima de un punto. De esta última forma se abrirá un menú contextual donde el usuario podrá seleccionar la acción *Eliminar*. También se pueden mover los puntos seleccionados si se hace clic con el botón izquierdo del ratón encima de un punto seleccionado y manteniendo la pulsación se mueve el puntero por el visor. Desde *Selección (Edición)* también se pueden crear puntos delimitadores. Para ello, el usuario debe hacer clic derecho del ratón en las proximidades de una arista que conecte dos puntos delimitadores.

Se abrirá un menú contextual y el usuario podrá seleccionar *Añadir*. Al pulsar en *Añadir* se creará un punto nuevo en la posición del ratón, y los puntos que conectaba la arista pasarán a ser los vecinos del nuevo punto creado.

La agrupación de puntos delimitadores constituyen un contorno. Un contorno segmenta de forma válida una parte del cometa si está compuesto como mínimo por tres puntos delimitadores. Además, cada punto delimitador del contorno tiene exactamente dos vecinos, o lo que es lo mismo, está conectado a otros dos puntos del mismo contorno. Estas conexiones son recíprocas. Es decir, si un punto está conectado con otro punto, este otro punto lo está también con el primero. Un punto delimitador no se puede conectar consigo mismo. Tampoco se puede conectar a puntos de diferente tipo o a puntos que ya pertenecen a un contorno cerrado. Estas restricciones hacen que un contorno cerrado se pueda ver como un grafo no dirigido, conexo y en el que para cada uno de sus nodos existen únicamente dos caminos donde el nodo origen es también el nodo destino.

Cada vez que el usuario conecta dos puntos delimitadores se comprueba si en el contorno al que pertenecen existe un camino cerrado. El sistema determina si este camino cerrado existe mediante el Algoritmo 5.1. El método recibe el contorno que se va a estudiar y uno de los puntos que participaron en la conexión, el cual actuará como nodo origen del grafo. El algoritmo viaja desde el nodo origen al resto de nodos del contorno siguiendo las conexiones establecidas entre estos. El algoritmo determina que existe un camino cerrado si es capaz de llegar al nodo origen a través de un camino de nodos sin repetir. Cuando en un contorno se encuentra un camino que lo cierra se eliminan los puntos del contorno que no pertenecen a este camino y el contorno se marca como *cerrado*.

Algoritmo 5.1: Método recursivo escrito en Python que determina si un contorno bajo edición/construcción está cerrado.

```
1
2 '''
3     Checks if a CanvasContour is closed.
4 '''
5 def _check_canvas_contour_is_closed(contour, root_point):
6
7     delimiter_point_id_list = []
8
9     for neighbor in root_point._get_neighbors():
10
11         delimiter_point_id_list = _union(
12             delimiter_point_id_list,
13             __check_contour_is_closed(
14                 neighbor,
15                 root_point._get_id(),
16                 root_point._get_id(),
```

```
                []
            ))

    contour._set_closed(
      root_point._get_id() in delimiter_point_id_list)
    return delimiter_point_id_list

'''
    Recursive function that returns a list with the DelimiterPoints
    that closes the CanvasContour.
'''
def __check_contour_is_closed(delimiter_point, sender_id, root_id,

    delimiter_point_id_list):

    if len(delimiter_point._get_neighbors()) < 2:
        return []

    # Point is root
    if delimiter_point._get_id() == root_id:
        return _union(delimiter_point_id_list.copy(), [root_id])

    for neighbor in delimiter_point._get_neighbors():

        # Neighbor isn't sender
        if neighbor._get_id() != sender_id:

            delimiter_point_id_list = _union(
                delimiter_point_id_list,
                __check_contour_is_closed(
                    neighbor,
                    delimiter_point._get_id(),
                    root_id,
                    []
                )
            )

    if len(delimiter_point_id_list) > 0:
        return _union(delimiter_point_id_list,
                  [delimiter_point._get_id()]
             )

    return []
```

En la Figura 5.21 se adjunta un ejemplo con los efectos de aplicar el algoritmo. En la imagen de la izquierda se puede ver el estado del contorno inicial. En la imagen de la derecha

se puede ver el resultado de aplicar el algoritmo tras conectar el punto 1 con el punto 2.

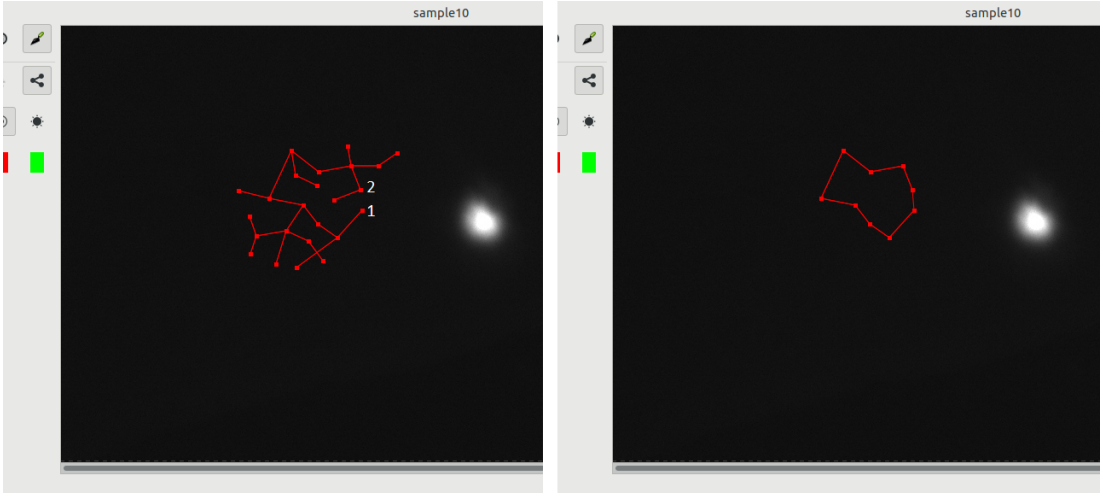


Figura 5.21: Resultado de conectar dos puntos que crean un camino cerrado en un contorno. A la izquierda, el contorno antes de la conexión. A la derecha, el contorno cerrado resultante tras conectar los puntos delimitadores 1 y 2.

Para crear un cometa con cabeza y cola primero se debe crear el contorno de la cola. A continuación, es necesario crear el contorno de la cabeza con al menos un punto delimitador en el interior o borde del contorno de la cola. Si por el contrario el contorno de la cabeza que se crea no está en contacto con ningún contorno de cola, el sistema creará un cometa compuesto únicamente por el contorno la cabeza. En la Figura 5.22 se muestra cómo responde el sistema bajo la creación de contornos en tres escenarios consecutivos. En la fila superior la aplicación está en modo *Edición* y se dibujan tanto los cometas registrados como válidos en el sistema como los contornos en construcción. En la fila inferior está en modo *Selección* y sólo se dibujan los cometas registrados como válidos. En la imagen inferior de la primera columna no se ve ningún cometa porque el contorno de la cola de la imagen superior, a pesar de ser un contorno válido, no es suficiente por si solo para formar un cometa. En la segunda columna se crea otro contorno de cola y un contorno de cabeza. El contorno de la cabeza no está en contacto con ningún contorno de cola por lo que se registra un cometa con él y de este modo se dibuja en la imagen inferior. En la tercera y última columna se crea un contorno de cabeza en el interior del primer contorno de cola que se creó, generándose así un cometa con cabeza y cola.

Los cometas válidos registrados en el sistema también se dibujan en el estado *Edición* pero no son editables. Para editar manualmente un cometa que ya ha sido registrado, el usuario deberá seleccionar el cometa desde el estado *Selección* y a continuación seleccionar *Editar cometa* desde *Ventana de selección*. En la imagen de la Figura 5.23 se muestra una imagen de la ventana principal cuando el usuario está editando un cometa ya validado. *Ventana de selección* informa al usuario del estado en el que se encuentra la aplicación para esa imagen.

Con el resto de imágenes se podrá trabajar con normalidad. La edición de ese cometa para esa imagen seguirá en curso hasta que se cancele la acción o se guarden los cambios. El botón *Guardar* no está activable si los contornos del cometa no son válidos.

Otros métodos más directos de edición se logran seleccionando las acciones *Eliminar cometa* para eliminar el cometa por completo y *Eliminar cola* para eliminar la cola del cometa desde *Ventana de selección.*

Otra parte importante de la segmentación manual es la herramienta de ampliación. Esta se encuentra a la derecha del botón que genera la hoja de cálculo. El primer y último botón se utilizan para alejar y acercar la imagen en el visor, respectivamente. El cuadro combinado del centro permite la selección de un nivel de ampliación concreto o la introducción al usuario de un nivel arbitrario. Los nuevos niveles de ampliación introducidos por el usuario que no pertenecen al modelo del cuadro combinado se registran. Cada imagen tiene un modelo de ampliación propio.

Incrementando la ampliación en las imágenes lo hace también el detalle con el que se pueden definir los contornos de los cometas. Por defecto, los componentes de GTK renderizaban por completo las imágenes a pesar de que sólo un área de esta era visible a través del visor. Esto hacía que a medida que se ampliaban las imágenes, la renderización de cada *frame* fuera más costosa, hasta hacer que la aplicación fuera inservible. Por esta razón, se implementó que cada vez que el visor tuviera que redibujar la imagen únicamente lo hiciera del área que fuera a ser visible.

5.4 Pruebas

Para validar el *software* se realizaron tanto pruebas de unidad como de aceptación. Las pruebas de unidad se realizaron de forma manual al mismo tiempo que se iba desarrollando la aplicación para verificar los componentes implementados.

Para las pruebas de aceptación, el equipo se reunió con en cliente en una fecha fijada. Se presentó la aplicación y se comprobó el funcionamiento de todos los casos de uso. El cliente verificó el cumplimiento de los objetivos iniciales y dio el visto bueno de la aplicación.

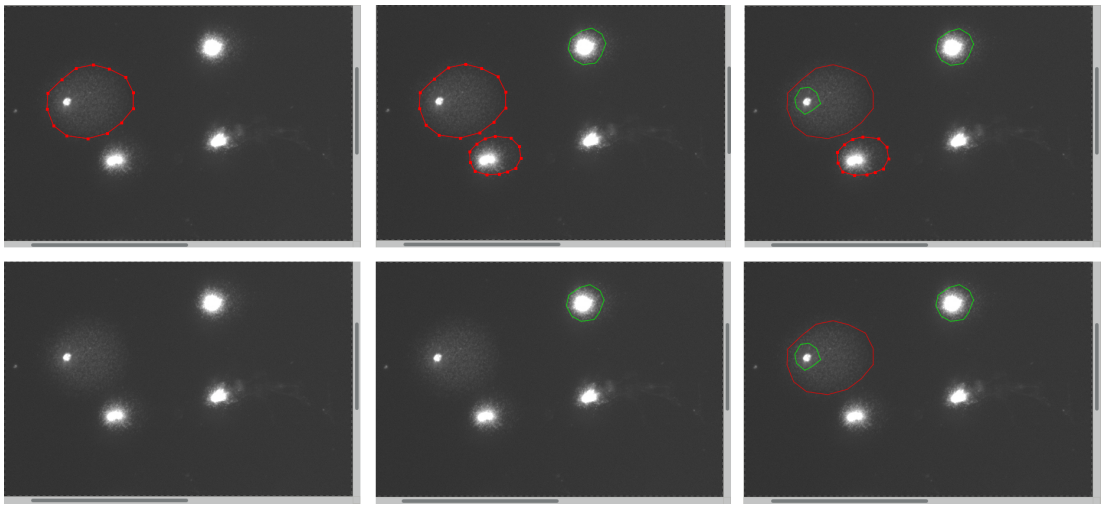


Figura 5.22: Ejemplo de creación de cometas mediante la segmentación manual. Los cambios se añaden de izquierda a derecha. En la fila de imágenes superior se muestra el sistema en el estado *Edición*. En la fila de imágenes inferior, el sistema se encuentra en el estado *Selección*. En la primera columna se crea un contorno de cola. En la segunda columna se crea otro contorno de cola y otro contorno de cabeza. En la tercera y última columna se crea un contorno de cabeza en el interior del primer contorno creado de cola.

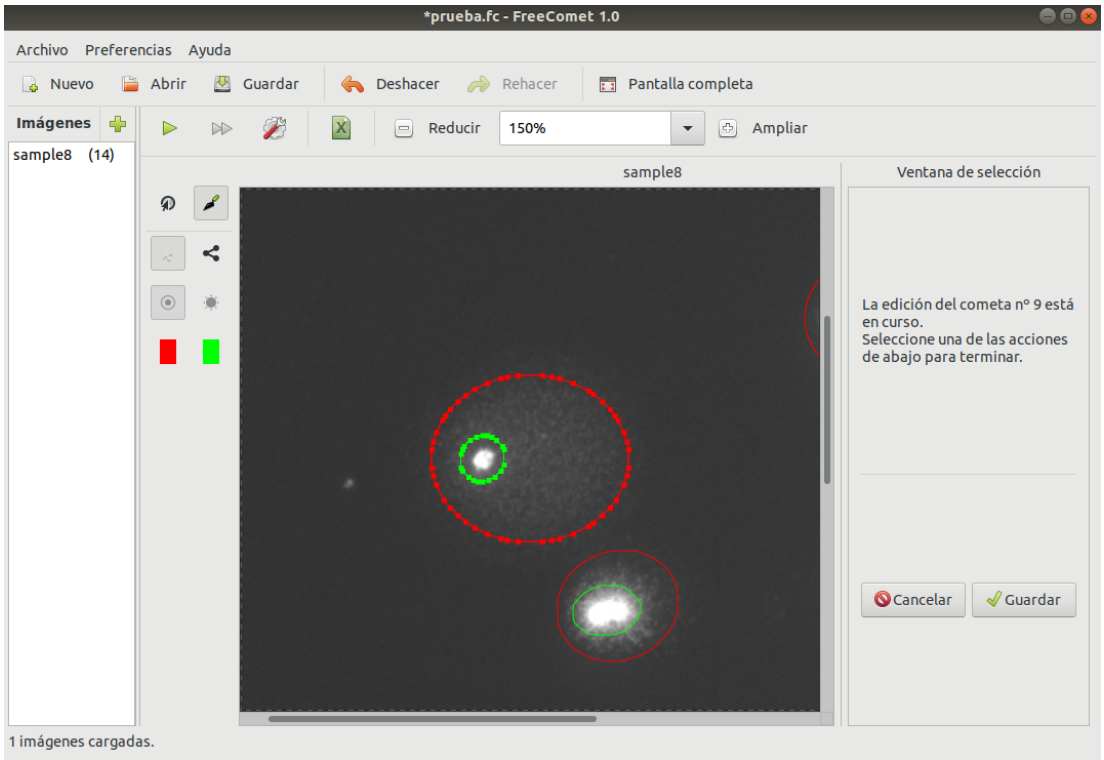


Figura 5.23: Ventana principal de la aplicación con la edición de un cometa activa.

Capítulo 6

Conclusiones y líneas futuras

La finalidad de este trabajo era ayudar al equipo de biólogos con la obtención de las métricas de sus imágenes del *ensayo del cometa*. Para ello se investigó sobre el ensayo, se profundizó en su estado del arte y se estudiaron las herramientas disponibles para el análisis de las imágenes. Se decidió desarrollar un nuevo algoritmo de procesado de imágenes que proporcionase mejores resultados que el resto de metodologías disponibles sobre el banco de imágenes en posesión. También se decidió integrar el algoritmo en una herramienta de segmentación manual. Con estos objetivos en mente se realizó el plan de proyecto.

Para validar el algoritmo propuesto se determinó que se compararían sus resultados con los de la metodología implementada en la herramienta *OpenComet*. Así, mediante el acceso a su documentación se implementó en Python y se decidió que se incluiría en el futuro sistema. Después y mediante el estudio del histograma de las imágenes, el empleo de métodos de umbralización, binarización, filtros morfológicos, métodos geométricos y un árbol de decisión, se implementó el nuevo algoritmo. Para maximizar la efectividad del algoritmo se ajustaron sus parámetros de forma empírica y se compararon sus resultados con los de OpenComet. Los resultados del algoritmo propuesto fueron significativamente mayores con una tasa de detección de objetos del 97.33 % y un valor promedio de *IoU* del 70.4 %. El bajo valor de la tasa de *IoU* derivó de la alta variabilidad presente en las imágenes. Para combatirla se procedió con el desarrollo de la aplicación que incluiría los algoritmos de procesado y la herramienta de segmentación manual.

Tras un análisis de los requisitos funcionales y no funcionales, descripción de casos de uso, diseños de la interfaz y código, se desarrolló una aplicación en un entorno de escritorio GTK que permitió tanto la segmentación automática como manual de las imágenes. Los principales objetivos del proyecto se vieron cumplidos y el equipo dio el visto bueno a la aplicación.

A pesar de haber cumplido con los objetivos principales del proyecto y haber añadido funcionalidades no primordiales, son muchas las mejoras que aún se pueden añadir. La implementación del algoritmo OpenComet en Python es lenta porque sus métodos no están

optimizados en este lenguaje. El algoritmo propuesto también está abierto a mejoras pues depende del tamaño de los cometas en las imágenes. Sería interesante implementar un nuevo algoritmo con imágenes del ensayo del cometa de diversas fuentes. Con respecto a la herramienta de segmentación manual, es sencilla y aporta una funcionalidad inexistente en las herramientas encontradas. No obstante, sus funcionalidades se podrían mejorar como el caso de la selección de los puntos delimitadores, que no permite la selección de puntos no visibles en el visor. También se podrían incluir otros métodos de segmentación predeterminada como la colocación de contornos elípticos ya creados en la imagen.

Apéndices

Apéndice A

Casos de uso

En este apéndice se describen de forma más detallada los casos de uso que los usuarios pueden ejecutar en la aplicación. Para cada caso de uso se incluye una tabla con el actor que lo puede disparar, una breve descripción, las precondiciones, las secuencias normal y alternativas si las hubiere, las postcondiciones y el nivel de importancia de este.

De la Tabla A.1 a la Tabla A.12 se describen los casos de uso relacionados con la gestión de los proyectos y las imágenes. De la Tabla A.14 a la Tabla A.25 se describen los casos de uso que tienen que ver con la segmentación de las imágenes. Finalmente, en la Tabla A.26 y Tabla A.27 se describen los casos de uso relacionados con la obtención de las métricas de los cometas segmentados.

Tabla A.1: Caso de uso ”Crear proyecto”.

C1	Crear proyecto
Actor	Usuario
Descripción	Se crea un proyecto nuevo.
Precondición	-
Secuencia normal	1a. El usuario selecciona la acción ”Nuevo” en la barra de herramientas. 1b. El usuario selecciona la acción ”Nuevo” en la barra de menú.
Postcondición	1. La lista de imágenes está vacía. 2. Se actualiza la etiqueta del nombre del proyecto con el nombre establecido por defecto.
Importancia	Media

Tabla A.2: Caso de uso ”Abrir proyecto”.

C2	Abrir proyecto
Actor	Usuario
Descripción	Se abre un proyecto existente.
Precondición	-
Secuencia normal	1a. El usuario selecciona la acción ”Abrir” en la barra de herramientas. 1b. El usuario selecciona la acción ”Abrir” en la barra de menú. 2. Se abre una ventana diálogo de selección de archivos. 3. El usuario selecciona el proyecto que desea abrir y pulsa el botón ”Abrir”.
Postcondición	1. La lista de imágenes se actualiza con la lista de imágenes del proyecto. 2. Se actualiza la etiqueta del nombre del proyecto con el nombre del proyecto abierto.
Importancia	Media

Tabla A.3: Caso de uso ”Guardar proyecto”.

C3	Guardar proyecto
Actor	Usuario
Descripción	Se guarda el proyecto actual en memoria.
Precondición	-
Secuencia normal	1a. El usuario selecciona la acción ”Guardar” en la barra de herramientas. 1b. El usuario selecciona la acción ”Guardar” en la barra de menú.
Secuencia alternativa	1. El usuario selecciona la acción ”Guardar como” en la barra de menú. 2. Se abre una ventana diálogo de selección de archivos. 3a. El usuario selecciona el proyecto existente que desea sobrescribir. 3b. El usuario introduce el nombre con el que quiere guardar el proyecto. 4. El usuario pulsa en el botón ”Guardar”.
Postcondición	-
Importancia	Media

Tabla A.4: Caso de uso ”Añadir imagen”.

C4	Añadir imagen
Actor	Usuario
Descripción	Se añaden una o más imágenes a la lista de imágenes.
Precondición	-
Secuencia normal	1. El usuario pulsa en el botón ”Añadir imágenes”. 2. Se abre un diálogo de selección de archivos. 3. El usuario selecciona las imágenes que quiere añadir y pulsa en el botón ”Añadir”.
Secuencia alternativa	1. El usuario pulsa el botón derecho del ratón en un espacio vacío en la lista de imágenes. 2. Se abre un menú contextual. 3. El usuario selecciona la acción ”Añadir”. 4. Se abre un diálogo de selección de archivos. 5. El usuario selecciona las imágenes que quiere añadir y pulsa el botón ”Añadir”.
Postcondición	1. Las nuevas imágenes aparecen en la lista de imágenes.
Importancia	Alta

Tabla A.5: Caso de uso ”Eliminar imagen”.

C5	Eliminar imagen
Actor	Usuario
Descripción	Se elimina la imagen seleccionada.
Precondición	1. La lista de imágenes no está vacía. 2. Una imagen está seleccionada.
Secuencia normal	1. El usuario pulsa la tecla ”Supr”.
Secuencia alternativa	1. El usuario pulsa el botón derecho del ratón sobre una imagen en la lista de imágenes. 2. Se abre un menú contextual. 3. El usuario selecciona la acción ”Eliminar”
Postcondición	1. La imagen desaparece de la lista de imágenes.
Importancia	Media

Tabla A.6: Caso de uso ”Cambiar nombre de imagen”.

C6	Cambiar nombre de imagen
Actor	Usuario
Descripción	Se cambia el nombre de la imagen seleccionada.
Precondición	1. La lista de imágenes no está vacía.
Secuencia normal	1. El usuario pulsa el botón derecho del ratón sobre una imagen en la lista de imágenes. 2. Se abre un menú contextual. 3. El usuario selecciona la acción ”Cambiar nombre”. 4. El usuario introduce el texto con el que quiere renombrar la imagen. 5. El usuario pulsa la tecla ”Enter”.
Postcondición	-
Importancia	Baja

Tabla A.7: Caso de uso ”Activar modo *pantalla completa*”.

C7	Activar modo *pantalla completa*
Actor	Usuario
Descripción	La aplicación activa el modo *pantalla completa*.
Precondición	1. El modo pantalla completa está desactivado.
Secuencia normal	1. El usuario pulsa en el botón ”Pantalla completa” en la barra de herramientas.
Postcondición	-
Importancia	Baja

Tabla A.8: Caso de uso ”Desactivar modo *pantalla completa*”.

C8	Desactivar modo *pantalla completa*
Actor	Usuario
Descripción	La aplicación desactiva el modo *pantalla completa*.
Precondición	1. El modo pantalla completa está activado.
Secuencia normal	1. El usuario pulsa en el botón ”Pantalla completa” en la barra de herramientas.
Postcondición	-
Importancia	Baja

Tabla A.9: Caso de uso ”Rehacer última acción”.

C9	Rehacer última acción
Actor	Usuario
Descripción	La pila de acciones ejecutadas deshechas no está vacía.
Precondición	-
Secuencia normal	1. El usuario pulsa en el botón ”Rehacer” en la barra de herramientas.
Postcondición	-
Importancia	Baja

Tabla A.10: Caso de uso ”Deshacer última acción”.

C10	Deshacer última acción
Actor	Usuario
Descripción	La pila de acciones ejecutadas no está vacía.
Precondición	-
Secuencia normal	1. El usuario pulsa en el botón ”Deshacer” en la barra de herramientas.
Postcondición	-
Importancia	Baja

Tabla A.11: Caso de uso ”Cambiar idioma”.

C11	Cambiar idioma
Actor	Usuario
Descripción	Se traducen todos los textos de la aplicación al idioma elegido.
Precondición	-
Secuencia normal	1. El usuario pulsa en la pestaña ”Preferencias” en la barra de menú. 2. Se abre un menú desplegable. 3. El usuario pulsa en la pestaña ”Idioma”. 4. Se abre un submenú lateral. 5. El usuario pulsa en el idioma al que desea traducir la aplicación.
Postcondición	-
Importancia	Media

Tabla A.12: Caso de uso ”Voltear imagen horizontalmente”.

C12	Voltear imagen horizontalmente
Actor	Usuario
Descripción	Se voltean los píxeles de la imagen seleccionada en el eje horizontal.
Precondición	1. Una imagen está seleccionada. 2. El modo ”Selección” está activo.
Secuencia normal	1. El usuario pulsa el botón derecho del ratón en cualquier punto de la imagen en el visor de imágenes. 2. Se abre un menú contextual. 3. El usuario selecciona la acción ”Voltear horizontalmente”.
Postcondición	-
Importancia	Baja

Tabla A.13: Caso de uso ”Invertir color de imagen”.

C13	Invertir color de imagen
Actor	Usuario
Descripción	Se invierten los valores de intensidad de los píxeles de la imagen seleccionada.
Precondición	1. Una imagen está seleccionada. 2. El modo ”Selección” está activo.
Secuencia normal	1. El usuario pulsa el botón derecho del ratón en cualquier punto de la imagen en el visor de imágenes. 2. Se abre un menú contextual. 3. El usuario selecciona la acción ”Invertir color”.
Postcondición	-
Importancia	Baja

Tabla A.14: Caso de uso ”Segmentar imagen”.

C14	Segmentar imagen
Actor	Usuario
Descripción	Se segmentan una o más imágenes de la lista de imágenes.
Precondición	1. La lista de imágenes no está vacía.
Secuencia normal	1. El usuario pulsa en el botón ”Analizar muestras”. 2. Se abre una ventana de selección de imágenes con la lista de todas las imágenes. 3. El usuario marca las casillas de selección de las imágenes que quiere segmentar. 4. El usuario pulsa en el botón ”Siguiente”. 5. Se abre una ventana de configuración del algoritmo de segmentado. El usuario tiene la opción de escoger el algoritmo y configurar sus parámetros. 6. El usuario pulsa en el botón ”Ejecutar”. 7. Se abre una ventana de progreso del análisis. 8. Cuando el análisis termina, la ventana se cierra.
Secuencia alternativa (I)	1. El usuario pulsa el botón ”Análisis rápido”. 2. Se abre una ventana de progreso del análisis. 3. Cuando el análisis termina, la ventana se cierra.
Secuencia alternativa (II)	1. El usuario pulsa el botón derecho del ratón sobre la imagen de la lista de imágenes que quiere segmentar. 2. Se abre un menú contextual. 3. El usuario selecciona la acción ”Analizar”. 4. Se abre una ventana de configuración del algoritmo de segmentado. El usuario tiene la opción de escoger el algoritmo y configurar sus parámetros. 5. El usuario pulsa en el botón ”Ejecutar”. 6. Se abre una ventana de progreso del análisis. 7. Cuando el análisis termina, la ventana se cierra.
Postcondición	1. El número de cometas segmentados aparece para cada imagen segmentada en la lista de imágenes.
Importancia	Alta

Tabla A.15: Caso de uso ”Configurar algoritmo de segmentación”.

C15	Configurar algoritmo de segmentación
Actor	Usuario
Descripción	Se configura el algoritmo de segmentado automático de imágenes.
Precondición	-
Secuencia normal	1. El usuario pulsa en el botón ”Parámetros”. 2. Se abre una ventana de configuración del algoritmo de segmentado. 3. El usuario escoge el algoritmo de segmentación y configura sus parámetros. 4. El usuario pulsa en el botón ”Guardar”.
Postcondición	-
Importancia	Media

Tabla A.16: Caso de uso ”Añadir cometa”.

C16	Añadir cometa
Actor	Usuario
Descripción	Se añade un cometa nuevo a la imagen seleccionada.
Precondición	1. Una imagen está seleccionada. 2. El modo ”Edición” está activo.
Secuencia normal	1. El usuario crea el contorno cerrado de la cabeza en el visor de imágenes. 2. Se añade un cometa sin cola.
Secuencia alternativa	1. El usuario crea el contorno de la cola en el visor de imágenes. 2. El usuario crea el contorno de la cabeza en el interior del contorno del cometa. 3. Se añade un cometa con cabeza y cola.
Postcondición	1. La etiqueta con el número de cometas segmentados se incrementa.
Importancia	Alta

Tabla A.17: Caso de uso ”Eliminar cometa”.

C17	Eliminar cometa
Actor	Usuario
Descripción	Se elimina el cometa seleccionado.
Precondición	1. Un cometa está seleccionado.
Secuencia normal	1. En *Ventana de selección*, el usuario pulsa en el botón ”Eliminar cometa”.
Secuencia alternativa	1. En *Ventana de selección*, el usuario pulsa en el botón ”Editar cometa”. 2. El usuario selecciona y elimina todos los puntos delimitadores del cometa. 3. El usuario pulsa en el botón ”Guardar”.
Postcondición	1. La etiqueta con el número de cometas segmentados se decrementa.
Importancia	Alta

Tabla A.18: Caso de uso ”Editar cometa”.

C18	Editar cometa
Actor	Usuario
Descripción	Se editan los contornos del cometa seleccionado.
Precondición	1. Un cometa está seleccionado.
Secuencia normal	1. En *Ventana de selección*, el usuario pulsa en el botón ”Editar cometa”. 2. El usuario edita los contornos del cometa añadiendo, conectando, moviendo y eliminando puntos delimitadores. 3. El cometa tiene sólo contorno de la cabeza o tanto contorno de la cola como de la cabeza. 4. El usuario pulsa en el botón ”Guardar”.
Postcondición	-
Importancia	Alta

Tabla A.19: Caso de uso ”Añadir punto delimitador”.

C19	Añadir punto delimitador
Actor	Usuario
Descripción	Se añade un punto delimitador.
Precondición	1. Una imagen está seleccionada. 2. El modo ”Edición” está activo.
Secuencia normal	1. El usuario selecciona el modo ”Construcción”. 2. El usuario elige el tipo de contorno a construir. 3. El usuario pulsa con el botón izquierdo del ratón en el visor de imágenes.
Secuencia alternativa	1. El usuario selecciona el modo ”Selección (Edición)”. 2. El usuario pulsa con el botón derecho del ratón en una arista de un contorno. 3. Se abre un menú contextual. 4. El usuario selecciona la acción ”Añadir”.
Postcondición	-
Importancia	Alta

Tabla A.20: Caso de uso ”Conectar puntos delimitadores”.

C20	Conectar puntos delimitadores
Actor	Usuario
Descripción	Se conectan dos puntos delimitadores de contornos del mismo tipo.
Precondición	1. Una imagen está seleccionada. 2. El modo ”Construcción” está activo.
Secuencia normal	1. El usuario elige el tipo de contorno de los puntos. 2. El usuario selecciona un punto delimitador del tipo escogido. 3. El usuario pulsa el botón izquierdo del ratón en otro punto delimitador válido del mismo tipo.
Postcondición	1. Se dibuja una arista entre los dos puntos delimitadores.
Importancia	Alta

Tabla A.21: Caso de uso ”Mover puntos delimitadores”.

C21	Mover puntos delimitadores
Actor	Usuario
Descripción	Se mueven uno o más puntos delimitadores.
Precondición	1. Una imagen está seleccionada. 2. El modo ”Selección (edición)” está activo.
Secuencia normal	1. El usuario pulsa y mantiene el botón izquierdo del ratón sobre un punto delimitador 2. Con el botón mantenido pulsado, el usuario mueve el puntero del ratón al destino deseado de la imagen. 3. El usuario suelta el botón.
Postcondición	-
Importancia	Alta

Tabla A.22: Caso de uso ”Eliminar punto delimitador”.

C22	Eliminar punto delimitador
Actor	Usuario
Descripción	Se eliminan uno o más puntos delimitadores.
Precondición	1. Una imagen está seleccionada. 2. El modo ”Selección (edición)” está activo.
Secuencia normal	1. El usuario selecciona uno o más puntos delimitadores. 2. El usuario pulsa la tecla ”Supr”.
Secuencia alternativa	1. El usuario pulsa el botón derecho sobre un punto delimitador. 2. Se abre un menú contextual. 3. El usuario selecciona la acción ”Eliminar”.
Postcondición	-
Importancia	Alta

Tabla A.23: Caso de uso ”Eliminar cola”.

C23	Eliminar cola
Actor	Usuario
Descripción	Se elimina la cola del cometa seleccionado.
Precondición	1. Un cometa está seleccionado. 2. El cometa tiene cola.
Secuencia normal	1. En *Ventana de selección*, el usuario pulsa en el botón ”Eliminar cola”.
Secuencia alternativa	1. En *Ventana de selección*, el usuario pulsa en el botón ”Editar cometa”. 2. El usuario elimina todos los puntos delimitadores del contorno de la cola. 3. El usuario pulsa en el botón ”Guardar”.
Postcondición	-
Importancia	Baja

Tabla A.24: Caso de uso ”Acercar imagen”.

C24	Acercar imagen
Actor	Usuario
Descripción	Se amplía visualmente la imagen seleccionada.
Precondición	1. Una imagen está seleccionada.
Secuencia normal	1. El usuario pulsa en el botón ”Ampliar”.
Secuencia alternativa (I)	1. El usuario abre el cuadro combinado de ampliación. 2. El usuario selecciona un nivel de ampliación superior al actual.
Secuencia alternativa (II)	1. El usuario pulsa en la entrada de texto del cuadro combinado de ampliación. 2. El usuario introduce un nivel de ampliación superior al actual. 3. El usuario pulsa la tecla ”Enter”.
Postcondición	-
Importancia	Alta

Tabla A.25: Caso de uso ”Alejar imagen”.

C25	Alejar imagen
Actor	Usuario
Descripción	Se reduce visualmente la imagen seleccionada.
Precondición	1. Una imagen está seleccionada.
Secuencia normal	1. El usuario selecciona la imagen de la lista de imágenes que quiere reducir. 2. El usuario pulsa en el botón ”Reducir”.
Secuencia alternativa (I)	1. El usuario abre el cuadro combinado de ampliación. 2. El usuario selecciona un nivel de ampliación inferior al actual.
Secuencia alternativa (II)	1. El usuario pulsa en la entrada de texto del cuadro combinado de ampliación. 2. El usuario introduce un nivel de ampliación inferior al actual. 3. El usuario pulsa la tecla ”Enter”.
Postcondición	-
Importancia	Alta

Tabla A.26: Caso de uso ”Generar hoja de cálculo”.

C26	Generar hoja de cálculo
Actor	Usuario
Descripción	Se genera una hoja de cálculo con las métricas de los cometas de todas las imágenes segmentadas.
Precondición	La lista de imágenes no está vacía.
Secuencia normal	1. El usuario pulsa en el botón ”Generar archivo excel” en la barra de herramientas. 2. Se abre un diálogo de selección de archivos. 3a. El usuario selecciona la hoja de cálculo existente que desea sobrescribir. 3b. El usuario introduce el nombre con el que quiere guardar la nueva hoja de cálculo. 4. El usuario pulsa en el botón ”Guardar”.
Postcondición	-
Importancia	Alta

Tabla A.27: Caso de uso ”Ver estadísticas de cometa”.

C27	Ver estadísticas de cometa
Actor	Usuario
Descripción	Se abre una ventana con las métricas del cometa seleccionado.
Precondición	1. Un cometa está seleccionado.
Secuencia normal	1. En *Ventana de selección*, el usuario pulsa en el botón ”Ver parámetros”.
Postcondición	-
Importancia	Media

Lista de acrónimos

ADN Ácido desoxirribonucleico.

CRUD *Create, Read, Update, Delete.*

GDK *Graphics Drawing Kit.*

GTK *The GIMP Toolkit.*

IoU *Intersection over Union.*

TIFF *Tagged Image File Format*

UML *Unified Modeling Language.*

Glosario

Algoritmo Palabra que viene del nombre del matemático árabe Al-Khwarizmi (780 - 850 aprox.). Define el conjunto de instrucciones que sirven para ejecutar una tarea o resolver un problema. Los motores de búsqueda usan algoritmos para mostrar los resultados de búsquedas.

Aplicación Cualquier programa que corra en un sistema operativo y que haga una función específica para un usuario. Por ejemplo, procesadores de palabras, bases de datos, agendas electrónicas, etc.

Bit Dígito Binario. Unidad mínima de almacenamiento de la información cuyo valor puede ser 0 ó 1 (falso o verdadero respectivamente). Hay 8 bits en un *byte*.

Código fuente Conjunto de instrucciones que componen un programa, escrito en cualquier lenguaje. En inglés se dice *source code*.

Interfaz gráfica de usuario En inglés Graphic User Interface, corto como GUI. Componente de una aplicación informática que el usuario visualiza gráficamente, y a través de la cual opera con ella. Está formada por ventanas, botones, menús e iconos, entre otros elementos (p. ej. Mac OS, Windows y X window).

Multiplataforma Dicho de una aplicación o de un producto informático: Que puede ser utilizado por distintos sistemas o entornos.

Píxel El componente más pequeño y discreto de una imagen en un monitor o pantalla. Es un punto mínimo. Mientras mayor es el número de píxeles por pulgada, mayor es la resolución.

Ratón (*Mouse*) Dispositivo electrónico de pequeño tamaño operable con la mano y mediante el cual se pueden dar instrucciones a la computadora para que lleve a cabo una determinada acción.

Software Se refiere a programas en general, aplicaciones, juegos, sistemas operativos, utilitarios, antivirus, etc. Lo que se pueda ejecutar en la computadora.

Windows Sistema operativo desarrollado por la empresa Microsoft.

Bibliografía

[1] A. Rodríguez-Rey, E. Noris-García, and M. Torres, "Principios y relevancia del ensayo cometa," *Revista Cubana de Investigaciones Biomédicas*, vol. 35, pp. 184–194, 06 2016.

[2] O. Ostling and K. Johanson, "Microelectrophoretic study of radiation-induced dna damages in individual mammalian cells," *Biochemical and Biophysical Research Communications*, vol. 123, pp. 291–298, 08 1984.

[3] R. R. Narendra P.Singh, Michael T.McCoy and E. L.Schneider, "A simple technique for quantitation of low levels of dna damage in individual cells," *Cell Biology and Toxicology*, vol. 175, pp. 184–191, 03 1988.

[4] T. Kumaravel, B. Vilhar, S. Faux, and A. Jha, "Comet assay measurements: A perspective," *Cell biology and toxicology*, vol. 25, pp. 53–64, 12 2007.

[5] A. Jha, "Ecotoxicological applications and significance of the comet assay," *Mutagenesis*, vol. 23, pp. 207–21, 06 2008.

[6] M. Dusinska and A. Collins, "The comet assay in human biomonitoring: Gene-environment interactions," *Mutagenesis*, vol. 23, pp. 191–205, 06 2008.

[7] P. Olive, "Olive plimpact of the comet assay in radiobiology. mutat res 681: 13-23," *Mutation research*, vol. 681, pp. 13–23, 12 2007.

[8] G. Wasson, V. McKelvey-Martin, and C. Downes, "The use of the comet assay in the study of human nutrition and cancer," *Mutagenesis*, vol. 23, pp. 153–62, 06 2008.

[9] D. Mckenna, S. McKeown, and V. McKelvey-Martin, "Potential use of the comet assay in the clinical management of cancer," *Mutagenesis*, vol. 23, pp. 183–90, 06 2008.

[10] A. Wojcik. (2003) Casplab. [En línea]. Disponible en: http://casplab.com/

[11] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, 4th Edition.* Tennessee, EEUU: Pearson, 2017.

[12] T. Corp. (2017) Cometscore. [En línea]. Disponible en: http://rexhoover.com/

[13] Instem. (2020) Comet assay iv. [En línea]. Disponible en: https://www.instem.com/solutions/genetic-toxicology/comet-assay.php

[14] Andor. (2020) Komet 7. [En línea]. Disponible en: https://andor.oxinst.cn/assets/uploads/products/andor/documents/Komet-Brochure.pdf

[15] B. M. Gyori and G. Venkatachalam. (2020) Opencomet. [En línea]. Disponible en: http://www.cometbio.org/

[16] K. Schwaber and J. Sutherland. (2013) Scrum. [En línea]. Disponible en: https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf

[17] A. Zuo, J. Yang, and X. Chen, "Research of agile software development based on formal methods," 12 2010, pp. 762 – 766.

[18] Indeed. (2019) Salario promedio de un investigador en españa. [En línea]. Disponible en: https://www.indeed.es/salaries/investigador-Salaries

[19] ——. (2020) Salario promedio de un programador junior en españa. [En línea]. Disponible en: https://www.indeed.es/salaries/programador-junior-Salaries

[20] ——. (2020) Salario promedio de un programador analista en españa. [En línea]. Disponible en: https://www.indeed.es/salaries/analista-programador-Salaries

[21] ——. (2020) Salario promedio de un jefe de proyecto en españa. [En línea]. Disponible en: https://www.indeed.es/salaries/jefe-de-proyecto-Salaries

[22] G. van Rossum. (2020) Python. [En línea]. Disponible en: https://www.python.org/

[23] N. developers. (2020) Numpy. [En línea]. Disponible en: https://numpy.org/

[24] O. team. (2019) Opencv. [En línea]. Disponible en: https://opencv.org/

[25] S. van der Walt, J. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. Warner, N. Yager, E. Gouillart, T. Yu, and contributors, "scikit-image: Image processing in python," *PeerJ*, vol. 2, 07 2014.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, and G. Louppe, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, 01 2012.

[27] G. Team. (2020) Gtk. [En línea]. Disponible en: https://www.gtk.org/

[28] T. G. Project. (2020) Gdk. [En línea]. Disponible en: https://developer.gnome.org/gdk3/stable/

[29] C. W. Keith Packard and B. Esfahbod. (2019) Pycairo. [En línea]. Disponible en: https://www.cairographics.org/pycairo/

[30] T. G. Project. (2020) Gdk-pixbuf. [En línea]. Disponible en: https://developer.gnome.org/gdk-pixbuf/

[31] ——. (2020) Glib. [En línea]. Disponible en: https://developer.gnome.org/glib/

[32] G. Project. (2019) gettext. [En línea]. Disponible en: https://www.gnu.org/software/gettext/

[33] T. G. Foundation. (2018) Glade. [En línea]. Disponible en: https://glade.gnome.org/

[34] G. Project. (2014) Dia. [En línea]. Disponible en: http://dia-installer.de

[35] A. Thomas and D. Barashev. (2019) Ganttproject. [En línea]. Disponible en: https://www.ganttproject.biz/

[36] K. team. (2016) Kolourpaint. [En línea]. Disponible en: http://kolourpaint.sourceforge.net/

[37] M. Windows. (2020) Microsoft paint. [En línea]. Disponible en: https://support.microsoft.com/es-es/help/4027344/windows-10-get-microsoft-paint

[38] Adobe. (2020) Adobe photoshop touch. [En línea]. Disponible en: https://www.adobe.com/es/products/photoshop.html

[39] P. Brachet. (2020) Texmaker. [En línea]. Disponible en: https://www.xm1math.net/texmaker/

[40] W. E. R. G. W. Zack and S. A. Latt, "Automatic measurement of sister chromatid exchange frequency." *Journal of Histochemistry and Cytochemistry*, vol. 25 7, pp. 741–753, 07 1977.

[41] L.-K. Huang and M.-J. J. Wang, "Image thresholding by minimizing the measures of fuzzines," *Pattern Recognition*, vol. 28, pp. 41–51, 01 1995.

[42] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, 01 1979.

[43] L. Rokach and O. Maimon, "Decision trees," *The Data Mining and Knowledge Discovery Handbook*, vol. 6, pp. 165–192, 01 2005.

[44] T. Ayodele, *Introduction to Machine Learning*, 02 2010.

[45] E. Freeman, E. Freeman, B. Bates, and K. Sierra, *Head First Design Patterns*. O' Reilly and Associates, Inc., 2004.

[46] H. V. Björn Hellman and B. Boströmc, "The concepts of tail moment and tail inertia in the single cell gel electrophoresis assay," *Mutation Research/DNA Repair*, vol. 336, pp. 123–131, 03 1995.

[47] P. Olive, J. Banáth, and R. Durand, "Heterogeneity in radiation-induced dna damage and repair in tumor and normal cells measured using the "comet" assay," *Radiation research*, vol. 122, pp. 86–94, 05 1990.